University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: __3______

Date: _Mar 09, 2025___________

Group Number: ____110______

Name	**Student Number**	**CS Alias (Userid)**	**Preferred E-mail Address**
Frank Yang	11753753	j6l4t	ffyang@student.ubc.ca
Xingyang Zheng	57446361	c5i2r	xingyang2027@gmail.com
Shiyu Zhou	27214782	z9y5k	szhou49@outlook.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Project Summary:

This project focuses on **residential parking management**, addressing inefficiencies in traditional visitor pass systems. It ensures fair usage of visitor parking spaces by tracking parking activities, enforcing regulations, and preventing misuse. The system also provides administrators with tools to monitor violations, manage payments, and take enforcement actions when necessary.

Timeline and Task Breakdown

Remaining Tasks and Assignments

Due Date	Task	Assigned To
Mar 08	**2. Weekly Review Meeting #1**	Frank, Xingyang, Shiyu
Mar 09	1. Familiarize provided Documentations.	Frank, Xingyang, Shiyu
Mar 15	3.1 User Interface (UI) Design	Frank
Mar 15	3.2. Home Page & Authentication	Frank
Mar 15	3.3. Resident & Visitor Management	Frank
Mar 15	4.1 Database Design & Implementation	Xingyang
Mar 15	4.2 API Development & CRUD Operations	Shiyu

Mar 15	4.3 Queries & Database Operations	Xingyang
Mar 15	3.4. Parking Lot Overview & Real-Time Monitoring	Frank
Mar 15	**2. Weekly Review Meeting #2**	Frank, Xingyang, Shiyu
Mar 22	3.5. Violation Enforcement & Ticketing	Frank
Mar 22	3.6. Financial Tracking & Reports	Frank
Mar 22	3.7. Admin Settings & Logs	Frank
Mar 22	4.4 Violation Enforcement & Ticketing System	Shiyu
Mar 22	4.5 Real-Time Monitoring & Reporting	Xingyang
Mar 22	4.6 Role-Based Access Control	Shiyu
Mar 22	4.7 Financial Tracking	Xingyang
Mar 22	**2. Weekly Review Meeting #3**	Frank, Xingyang, Shiyu
Mar 29	4.8 API Connection & Integration	Shiyu
Mar 29	5.1 Setup Backend Server	Xingyang
Mar 29	5.2 Frontend Deployment	Frank

Mar 29	5.3 Manual API Testing	Shiyu
Mar 29	5.4 Database Validation	Xingyang
Mar 29	5.5 Test End-to-End User Flows	Shiyu
Mar 29	5.6 Check API & Frontend Deployment	Xingyang
Mar 29	**2. Weekly Review Meeting #4**	Frank, Xingyang, Shiyu
Apr 01	5.7 Final User Testing	Shiyu
Apr 01	6. Final review	Frank, Xingyang, Shiyu

Task Descriptions and Detailed Assignments

1. Familiarize provided Documentations.

- ☐ Familiarize provided documentations with JavaScript, Oracle SQL, Node.js, etc.
- ☐ Resources:
    - ☐ UBC CS-304 JavaScript and Oracle Guide
    - ☐ UBC CS-304 Project Overview
    - ☐ HTML Setup Guide

2. Weekly Review Meeting

- ☐ Review the progress and refine any incomplete sections.
- ☐ Ensure all parts of the project integrate smoothly.
- ☐ Debug and optimize code.
- ☐ Address any concerns or issues raised by team members.

3. Frontend Web Design Tasks

3.1 User Interface (UI) Design

- ☐ Design a modern, responsive user interface.
- ☐ Ensure clean navigation with a sidebar/dashboard layout for administrators and a simple menu for residents/visitors.

3.2. Home Page & Authentication

- ☐ Create a home page with a brief system overview and login options.
- ☐ Implement secure authentication (login, register, password reset) with role-based access (Admin, Resident, Visitor).

3.3. Resident & Visitor Management

- ☐ Resident Dashboard:

    Show assigned visitor pass quota and usage history.

    Allow residents to request visitor passes and see approval status.

- ☐ Visitor Registration Page:

    Allow visitors to register their vehicles and apply for a temporary pass.

3.4. Parking Lot Overview & Real-Time Monitoring

- ☐ Admin Dashboard:

    Display real-time parking lot occupancy with visual indicators for available, occupied, and violation spaces.

3.5. Violation Enforcement & Ticketing

Violation Reporting:

- [ ] Allow admins to manually enter vehicle details to check for unauthorized parking.
- [ ] Provide options to issue warning notices, penalties, or escalate to towing.

Ticket Management Page:

- [ ] Show a list of issued tickets with payment status.
- [ ] Enable online payment processing for fines.

3.6. Financial Tracking & Reports

- [ ] Payment Portal:
    - [ ] Secure page for users to pay fines online via credit card.
- [ ] Reports Section:
    - [ ] Generate real-time reports on revenue from fines, unpaid violations.
    - [ ] Generate violations report.

3.7. Admin Settings & Logs

- [ ] Admin panel to manage parking policies, set pass limits, and update system settings.
- [ ] View audit logs for issued visitor passes, payments, and rule enforcement actions.

4. Backend Development Tasks

4. 1 Database Design & Implementation

- [x] Design a relational database schema that models residents, visitors, parking lots, visitor passes, violations, and payments.
- [ ] Implement an SQL initialization script that:
    - [ ] Creates all tables and constraints.
    - [ ] Populates the database with sufficient data for testing.
    - [ ] Supports re-execution by including `DROP TABLE` statements.

4.2 API Development & CRUD Operations

- [ ] Develop a RESTful API to interact with the database.
- [ ] Implement core CRUD functionalities:
    - [ ] Residents & Visitors: Register new users, update profiles, retrieve visitor pass history.

  - ☐ Parking Management: Issue, update, and delete visitor passes.
  - ☐ Violation Handling: Add, modify, and remove parking violations.
  - ☐ Payment Processing: Log ticket payments and retrieve financial summaries.
- ☐ Input validation & sanitization to prevent SQL injection.

4.3 Queries & Database Operations

Implement at least 10 distinct queries covering:

- ☐ Insert Query: Allow users to register vehicles and request visitor passes. Ensure foreign key constraints are handled (e.g., reject if resident ID is invalid).
- ☐ Update Query: Enable users to update vehicle information while enforcing constraints (e.g., license plate uniqueness).
- ☐ Delete Query: Allow administrators to revoke visitor passes and cascade-delete violations if necessary.
- ☐ Selection Query: Enable filtering of parking records based on date, resident ID, or vehicle status.
- ☐ Projection Query: Let users view selected attributes (e.g., parking violations per resident without revealing sensitive data).
- ☐ Join Query: Retrieve visitor pass history by joining visitor, vehicle, and parking records.
- ☐ Aggregation with GROUP BY: Show the total number of visitor passes issued per resident.
- ☐ Aggregation with HAVING: Display residents who have received more than three parking violations.
- ☐ Nested Aggregation with GROUP BY: Find the average duration of visitor stays for each parking lot.
- ☐ Division Query: Identify residents who have used all available visitor passes in a given period.

4.4 Violation Enforcement & Ticketing System

- ☐ Implement a three-strike system to track repeat offenders.
- ☐ Provide an API to issue digital tickets and log payments.

4.5 Real-Time Monitoring & Reporting

- ☐ Develop APIs for live parking updates.
- ☐ Implement an admin dashboard API that retrieves:
  - ☐ Current parking lot occupancy.

    - [ ] Vehicles flagged for violation.

4.6 Role-Based Access Control

Implement role-based access:

  - [ ] Admin: Full control over parking enforcement, reports, and policy updates.
  - [ ] Resident: Can manage visitor passes, check personal violation records.
  - [ ] Visitor: Can register a vehicle and check parking status.

4.7 Financial Tracking

- [ ] Maintain a financial ledger tracking payments and outstanding dues.
- [ ] Generate reports.

4.8 API Connection & Integration

- [ ] Integrate Key API Endpoints
  - [ ] Residents & Visitors: Register, login, and manage visitor passes.
  - [ ] Parking Management: Request, approve, and revoke visitor passes.
  - [ ] Violation Handling: List violations, issue tickets, and process fines.
  - [ ] Payments: Submit and verify fine payments.
- [ ] **Handle API Responses & Errors**
  - [ ] Show success notifications for successful actions (e.g., "Visitor pass issued").
  - [ ] Handle API errors (e.g., unauthorized access, validation errors) with proper UI feedback.
  - [ ] Implement loading states for API calls.

5. Deployment, and Testing Tasks

5.1 Setup Backend Server

- [ ] Deploy the backend as a on the CS department server.
- [ ] Configure environment variables (.env) for database connection and API keys.
- [ ] Set up PM2 to manage server processes.

5.2 Frontend Deployment

- ☐ Deploying the frontend on the Vercel platform or local machine depends on the time frame.
- ☐ **Ensure** the frontend build process correctly references the deployed backend API URL.

5.3 Manual API Testing

- ☐ Use Postman to verify API endpoints.
- ☐ Test CRUD operations: Create, Read, Update, Delete.
- ☐ Test query logic (e.g., valid visitor pass issuance, payment tracking).

5.4 Database Validation

- ☐ Check that all database changes (INSERT, UPDATE, DELETE) reflect correctly.
- ☐ Verify referential integrity (e.g., deleting a resident also removes related visitor passes).

5.5 Test End-to-End User Flows

- ☐ Create a visitor pass, check if it appears in the admin dashboard.
- ☐ Simulate a violation and check if a ticket is generated.
- ☐ Process a payment and verify that it updates the database.

5.6 Check API & Frontend Deployment

- ☐ Ensure frontend correctly fetches data from the deployed backend.

5.7 Final User Testing

- ☐ Ask non-developers to interact with the system and report usability issues.

6. Final Review

- ☐ Conduct a final assessment of the project.
- ☐ Prepare necessary documentation and presentation materials.
- ☐ Ensure all requirements are met before submission.
- ☐ Deliverables:
    - ☐ Fully working system.
    - ☐ Documentation for system usage.

- ☐ Submission of project deliverables.

Potential Challenges in Collaboration and Solutions

Task Allocation and Progress Management:

Different team members are responsible for various aspects of the project, which may lead to difficulties in keeping progress synchronized and integrating different components smoothly. To address this, we will hold weekly meetings to track progress, ensure alignment, and make necessary adjustments. Regular check-ins will help prevent bottlenecks and ensure that all parts of the system work together seamlessly.

Technical Difficulties:

Challenges may arise in areas such as database design, API development, and frontend-backend communication. Issues like query optimization, authentication security, and real-time updates require careful implementation. To overcome these challenges, we will research best practices, collaborate on problem-solving, and consult the TA or relevant documentation when needed. Open discussions and knowledge sharing among team members will help us tackle technical obstacles effectively.

Code Integration and Testing:

As multiple members contribute code, differences in coding styles and approaches may lead to conflicts, making integration difficult and increasing the risk of bugs. To mitigate this, we will follow a consistent coding standard, use Git for version control, and conduct frequent testing. Regular code reviews and debugging sessions will ensure that issues are detected and resolved early, preventing major problems during the final integration phase.

Time Management

Balancing coursework and project deadlines can be challenging, potentially causing delays in individual contributions. To manage time effectively, we will plan ahead, set clear deadlines for each task, and communicate any scheduling conflicts as early as possible. If any team member faces difficulties keeping up, we will redistribute tasks accordingly to maintain steady progress and ensure the project is completed on time.