Maintenance and Support Document

1. Software Maintenance Strategy

Author will maintain the system regularly, fixing bugs in a timely manner, releasing updated versions according to the versioning strategy and follow a clear roadmap for future improvements. Bug fixes are handled as soon as possible, while new features or changes are bundled into planned updates. The roadmap ensures we keep track of long-term goals such as adding a database, a GUI, or advanced business logic.

Maintenance Guidelines:

Follow the DRY principle (Don't Repeat Yourself). When adding new features, reuse existing utility functions (such as input validation in utilities.py) as much as possible to avoid writing repetitive code.

Accommodate Single Responsibility Principle. Ensure each class and method is responsible for a single function. For example, system.py handles business logic, while main.py handles user interaction.

Use clear variable naming. Use meaningful variable and function names to make the purpose of the code easy to understand.

2. Versioning Strategy

This project follows Semantic Versioning in the format of MAJOR.MINOR.PATCH.

MAJOR: Big changes that are not backward compatible (e.g., new database structure).

MINOR: New features that do not break old ones.

PATCH: Bug fixes or small improvements.

Changelog should be maintained using a CHANGELOG.md file that records Added / Changed / Fixed / Deprecated items for each release.Each release should include code, migration scripts, updated requirements, and the changelog.

Project versioning roadmap is used to carry out future updates, according to the three major paths:

Short term: Fix bugs and improve error handling.

Medium term: Switch from SQLite database to more powerful server-side solutions like MySQL/PostgreSQL.

Long term: Major releases will introduce a Graphical User Interface (GUI) to replace the command-line user interface we have at the moment.

3. Backward Compatibility Strategy

We aim to keep the system backward compatible so old data and features continue to work after updates. The project should remain its compatibility within a MINOR versioning. Command-line menus and options should remain stable.

The file seed_cars.csv is the system's initial data source. During functional testing, this file can be backed up. In need of adding new initial vehicles, please add new rows in the correct format: car_id, make, model, year, mileage, available_now, min_rent_days, max_rent_days, daily_rate, fuel_type.

API stability: Command-line menus and functions remain the same whenever possible, so existing users are not affected.

Rollback plan. If migration fails, restore the backup database and re-run the previous release.

Testing compatibility. Use existing test data and repeat the rental workflow after each upgrade to confirm no breakage.