 Project Background

* The reviewed system design is a portion of an overall SuperRent system. The user this section of the product is focused upon is the customer. No functionality for managers or employees will be discussed in this design document or this review. The document concentrates of four major areas of functionality: customer vehicle rental, vehicle return, point deduction, and insurance purchase.
  + Customer Vehicle Rental: After the customers provide their information, they can make a reservation and have the rental agreement with credit card payment information.
  + Vehicle Return: Customers need to provide the following information, time, odometer reading and whether the gas tank has been refilled.
  + Point Deduction: The reward system includes the rules about how to collect points and deduct points.
    - When a customer joins the club, he or she receives 500 points. After the initial points, club members gain 1 point for every $5 they spend.
    - To deduct the points, the club members can exchange 1,000 points for a one day rental of a premium or lower-ranking car. Alternatively, 1,500 points for a one day rental for a luxury car, SUV, van or truck.
  + Insurance Purchasing: The customer can decide whether they need to have insurance purchase for the whole rental period. The payment need to be made at the point that vehicle is returned.

 Areas focused in the System Requirement review

* Correctness
* Usability
* Integrity

 Agenda and Checklist used in the formal review of the System Requirement

* Agenda
  1. Review Group 2’s System Requirements document
  2. Discuss thoughts on document
     1. Establish and review SQA plan
  3. Make modifications if necessary
  4. Conduct Formal Requirements Review
     1. Correctness
     2. Usability
     3. Integrity
  5. Record results of review
  6. Analyze results of review
  7. Produce and compile recommendations based on analysis
  8. Draft Final Review Report
     1. Delegate sections to members of the review team
* Checklist

|  |  |
| --- | --- |
| **Correctness** | |
| How well does the project plan cover the software integrity requirements? | There is very little mention of contingency plans for software correctness in the requirements |
| Have acceptance criteria been established for the work product? | The reports’ Use Cases give brief expectations of what the software is capable of and what is the expected resulting output |
| How is purpose and scope clearly defined for this project? | The overall project purpose and scope were covered in the Use Case diagram. Though there could have been more elaboration on how these functionalities affect the system as a whole. |
| Are references to policies, directives, procedures, standards, and terminology provided? | Procedures were explained in the sequence and data flow diagrams. Directives and standards are established in the Use cases. In terms of policies and terminology, there were not any examples of a data dictionary to explain terms to non-experts and uninformed stakeholders. |
| Are there any and all constraints/limitations that can be identified? | No constraints or limitations were defined for this project. Instead, there are only brief mentions of functional requirements. |
| How well defined does the purpose explain how the scope of the project can handle and adjust for new requirements? | The project doesn’t appear to have room for any possible additional requirements. The project scope only explains what components the developers wanted to discuss during their project timeline. |
| What kinds of plans are specified for software components? | The plans only specify the components for customer vehicle rental, return, point deduction, and insurance purchase. |
| Can the project life cycle allow for newly introduced required components and how can this be accomplished? | There doesn’t appear to be any room for any new requirements or components in the project timeline due to the strict Scrum development cycle. |
| **Analysis and Recommendations** | The project doesn’t clearly define software correctness to implement the requirements; however, the class diagrams and CRC cards are consistent with one another. There should be more room made to allow for any new requirements |

|  |  |
| --- | --- |
| **Usability** | |
| Based on existing documentation/information, do you understand the system in the context of each of the views in the system engineering hierarchy? | Upon further review, this documentation seems specifically designed for expert users and does not allow for much room for novice end-users |
| Is system output and input adequately defined? | The Use Cases and sequence diagrams explain what is the expected resulting output |
| Have expert and novice modes of interaction been defined? | There is no specified novice or expert interactions. Only base user requirements have been defined |
| Have important interfaces to all system elements been described? | The class diagrams clearly describe the important interfaces of which the user can interact with. |
| Is the behaviour of the software consistent with the information it must process and the functions it must perform? | Use cases, sequence diagrams, and data flow diagrams map out the software process by explaining from the point where the user inputs their data, how the system utilises the information, and what is the resulting output |
| Has the UI been designed effectively with use cases in mind? | There are many prime examples of UI usage, but some functionalities aren't completely covered in terms of user stories. For instance, the points deduction system is hardly discussed and we aren't given a clear idea of how the points are used. |
| Have business requirements been met in the use cases? | Not entirely, the document only specifies the component functionalities for customer vehicle rental, return, point deduction, and insurance purchase. |
| Have all users been identified? | Yes, the component the reviewed document covers is primarily meant for explaining the commonly used functions of the customer end-user |
| **Analysis and Recommendations** | Software is capable of carrying out its’ defined functionalities. This is explained in the use case, sequence, and data flow diagrams. Should contain novice and expert interaction definitions are highly recommended. |

|  |  |
| --- | --- |
| **Integrity** | |
| Have all data objects been described? | Yes, the class diagrams cover most of what data object the project will handle along with their consistent naming conventions. |
| Have all attributes been identified? | Most have been placed within the class diagrams, but there is little to none explanation on the usage and interactivity of these attributes. |
| Do the primary functions remain within scope and has each been adequately described? | The documentation refers to the system requirements to explain that this project will only work on the main functions of customer vehicle rental, vehicle return, point deduction, and insurance purchasing. |
| Does the system have a consistently designed user interface? | The menu content format is consistent across the different transition states, but there the navigation only seems to flow in one direction assuming that the user will not cancel their session or make any mistakes whatsoever. |
| Will end users be able to find the functions they are familiar with? | There is little mention and explanation from either the use cases or sequence diagrams on navigating functionalities. |
| Are proper naming conventions being followed? | There are inconsistencies within the variable naming of the elements, classes, and their attributes. |
| Do the developers’ goals match with the customers goals? | The aim of the development team is generally focused upon the main functions the customer will use. |
| Can the system prevent corruption? | There weren't any contingency plans made for the possibility of system corruption. |
| **Analysis and Recommendations** | The project doesn’t clearly define how software correctness will apply to the requirements and their implementation; however, the class diagram and CRC remain consistent. The project plan doesn’t consider accuracy or contingency as major requirements. |

 Results of the formal review of the System Requirement

* In the Vehicle Class, no specifications are placed and only references other tables are made. There is no requirement set to track vehicle beyond type. No date of purchase, no color, no requirements beyond a referencing other classes.
* The class structure seems strange to have a one-to-one relationship between carRentalReservation and Vehicle. This will prevent the vehicle from being available for other CarRentalReservations. To avoid this will involve deleting the CarRentalReservation entries to free up vehicles in the database as a requirement. The requirement to track everything but CarRentalReservation StartDate and confirmation number must be removed.
* DFD diagrams are described up to level 2 data flow, they are consistent and concise when describing correct states and defining transitions. Even though there is a lack of description, novices or uninformed users would appreciate a proper and full description accompanying the any of the diagrams.
* Nowhere in the user stories is it mentioned that a customer can use points to purchase rental days. Even though it is stated in the background that this is a requirement. The only requirement placed on the points system in the user stories, is to allow the customer to pay for their insurance with points. There should be user story for renting vehicles using the points system and then a requirement should be placed throughout the rest of the document so the final project implementation will not skip this crucial functionality.
* For the CarSalesAgreement class, the requirement to track the start and end time of a sales agreement is not shown anywhere in the customer requirements neither is it stated anywhere in the user stories. Tracking the odometer reading in that class shows off the lack of information stored in the vehicle class which has not attributes. Where is it coming from if it isn't coming from the vehicle itself?

 Areas focused in the System Design review

* Reusability
* Efficiency
* Flexibility

 Agenda and Checklist used in the formal review of the System Design

* Agenda
  1. Review Group 2’s System Design document
  2. Discuss thoughts on document
     1. Establish and review SQA plan
  3. Make modifications if necessary
  4. Conduct Formal Requirements Review
     1. Reusability
     2. Efficiency
     3. Flexibility
  5. Record results of review
  6. Analyze results of review
  7. Produce and compile recommendations based on analysis
  8. Draft Final Review Report
     1. Delegate sections to members of the review team
     2. Conduct peer reviews of report drafts
  9. Compile final draft of Final Review report
* Checklist

|  |  |
| --- | --- |
| **Reusability** | |
| What are the interfaces between the reused software and the operator? | The target system doesn’t interface between the plant system/environment and the reused software. |
| What are the interfaces between the reused software and the plant system/environment? | The target system doesn’t interface between the plant system/environment and the reused software. |
| How does the architecture define reusable software for the target and legacy systems? | The architecture well defines external legacy systems but doesn’t define reusable component of the legacy systems. |
| What method does the design define to reuse the software? | The design doesn’t define reuse method but each class can be reused for the system extension. |
| What environment must be initialized prior to executing the reused software? | The target system uses Web-based environment but it doesn’t initialize Web-browser UI. |
| What schema does the project adopt to reuse software components in the design phase? | The project document doesn’t define a reusable software schema. |
| Does each component in the design have specific, clear and well defined operations in each interface? | The component diagram doesn’t clearly define an interface for access. Each component designed very roughly and the interconnectivity is not very well-defined. |
| Is each component capable of handling repeated usage? | The components are not well designed for reuse but the class designs compensate for repeated usage. |
| **Analysis and Recommendations** | The design document doesn’t consider software reusability for the legacy and the target system. To improve the system quality and to reduce project cost, it is recommended that the project adopts software reuse strategy and technology. An especially good example we would recommend is RAS(Reusable Asset Specification) by the OMG(Object Management Group). These are a set of guidelines and recommendations about the structure, content, and descriptions of reusable software assets. |

|  |  |
| --- | --- |
| **Efficiency** | |
| Does the document contain data type information? | No, this leaves the inability to understand what each field will be used for and their capabilities. |
| Does the database cardinality function correctly with the user and efficiency requirements? | Database design does contain cardinality issues which cause major redundancies and general data structure flaws. |
| Are classes designed to reduce redundancies and minimize resource usage? | There are redundancies found within classes, functions, and attributes. Which cause increased resource usage and potential design errors. |
| Has an object-relationship model been defined? | Yes, the class diagram explains this quite briefly. |
| Is each subsystem appropriately allocated to processor and tasks? | Redundancies may cause overlap in tasks and processor usage. |
| How much memory is allocated to each component? | The amount of memory allocated isn’t explained in the plan as no data types are mentioned. |
| How is data communicated between software components? | The data flow diagram explains the process from which the user inputs information & commands, to the point where the system outputs its responses accordingly |
| Can simpler data structures be used? | No data types mentioned, redundancies, and improper naming conventions may cause overlapping and increased complexity to system implementation. |
| **Analysis and Recommendations** | Conflicting naming conventions cause various errors and overall general confusion. For instance, car type and vehicle type in separate classes cannot be properly linked to one another. |

|  |  |
| --- | --- |
| **Flexibility** | |
| Is the system capable of adapting to all types of problems or defects? | There are no contingency plans for handling any problems or defects. The project is assumed to be functioning perfectly throughout the development phase. |
| Is the system capable of allocating resources when under high stress? | The document only states normal circumstances of system usage. |
| Can the system retrieve information based on generalized queries/requests? | With each query there is a subclass and subtype to classify for the request of specific information. |
| Can this system evolve? | Apart from the main functions described in the project background, this system plans to incorporate other components for other types of users. |
| Can we adjust for any changes in requirements, internal or external, made by the customer? | The established plans are static and don't allow for any room to include new or changed requirements. |
| Can the system handle being introduced to new components? | The restrictive classes and lack of mentioned data types make it very difficult to introduce any new components. |
| Can the minimum and maximum information output be adjusted? | Without knowing what types of data the system uses, we cannot determine if the input or output can be adjusted at all. |
| What types of risks is the system capable of handling? | No contingency plans for faults or major system changes are made. |
| **Analysis and Recommendations** | The project does not seem very flexible when a major error or fault can occur. These documents were written on the basis that the system will run perfectly throughout the entire development and implementation phases. |

 Results of the formal review of the System Design

* Tracking the gas level in the vehicle table not in the rental records table seems redundant as all cars should be full upon leaving. The return fluctuation is based upon who rented it last; not on the specific car and specifically how to charge the customer on the current status of the vehicle. Therefore, it should be tied not to the car but to the renter.
* With the one-to-one cardinality between the Customer table and RentalSaleAgreement table a customer can only have one rental agreement ever. This means that the RentalSaleAgreement table must be deleted to allow for a customer to rent another car. Since a number of other tables reference the RentalSaleAgreement table this would leave them orphans with no connection to the costumer table. This is a major design flaw which would create major issues in implementation.
* Another one to one cardinality issues between VehicleType and the Vehicle table. Each vehicle type can only be attached to one vehicle. Which means there can be only one type of each vehicle in the database. This just makes having a separate table for type redundant as for every vehicle you would have to create a new type entry.
* No specific data types are mentioned which makes it difficult for the reviewers to understand the intended usage of data. Adding data types in the design document would show the user estimated memory usage and also how the information could be sorted and accessed.
* CarRentalReservation references a vehicle type. Vehicle type is not referenced as an attribute anywhere else in the class diagram. The attribute would seem to make sense to exist also in the vehicle class or at least one it is connected too otherwise we have no idea where the type is coming from.
* The Vehicle class appears to store a large number of functions with a small number of attributes with no id number referenced. This means that in the class there is no way to differentiate vehicles from each other based on an attribute it contains. A number of the functions placed inside the Vehicle class seem out of place attached directly to the vehicle. addInsurance() is also used in CarInsurance the question of why are there two addInsurance functions seem obvious as the tables are not connected directly to each other to show one may draw from the other. The deductPoint() function seems like it should be kept in the classes that handle payment not directly on the Vehicle. Other suspect functions include makeCashPayment() and makeCreditCardPayment() these functions though needed should be placed in a different class.

 Conclusion of the Overall Review

* The system requirements do not thoroughly describe design specifications. From what we have analyzed, the system design isn't flexible enough for future expansions. Further lack of consistency between requirements and design models must have more room for software reusability for the legacy and targeted system. For an effective project, it is highly recommended that the project be able to adopt document consistencies and software reusability practices.

 Feedback on your experience of having your project reviewed

* Group 1’s review was organized, thought full, and honest. They had a good set of questions that focused upon a number of issues with our documents’ content. The organization of the review was fairly structured and they had very flexible sections to allow for change. No point was pressed upon too hard and no critical or judgemental comments were made.
* They did not seem to completely understand a few of their questions, but that may have been due to not having enough notes. They did keep the review strictly focused towards the document. Most questions were well-worded and directly pointed to the document. Their questions were used to draw more information out about the document, the systems’ design, and to not criticize design choices.
* We accepted and applied their review recommendations in the areas they found lacking. Overall, the review was found to be very a positive experience. The reviewers were very effective in their task and we look forward to seeing the results of their final report.