**Project Summary Report**

The DriverPass project is an app requested by our client, DriverPass, who wants to be able to help students pass their driving test at the local DMV, as most attempts are unsuccessful. With the DriverPass app, students would have access to up-to-date online practice exams and on-the-road training. The DriverPass system is based on a three-tier system, consisting of the user's web browser on their desktop or mobile device, a web/app server (to handle business logic and serve web pages), and a database storage server.

When designing the documents and diagrams for this project, although we were not required to completely design every aspect of the DriverPass system, I made sure to be quite detailed and thorough in my approach and resulting deliverables. If given more time or instructed to do so, I would create many more UML Activity and Sequence Diagrams to define the system an associated component more comprehensively. After completing this assignment, I felt the System Design Document (SDD) was lacking, although I met all rubric (grading) requirements. So, I ended up drastically altering the SDD to add more detail, including a Table of Contents due to the magnitude of the added content. The SDD is not much more informative and provides more than what we needed for this project, which was initially only the Technical Requirements and a few diagrams.

One of the most important aspects in designing a system is to determine the users and their associated functionalities. While going over the requirements and scheduling meeting held by DriverPass, I jotted down as much relevant information as possible and divided the requirements into user requirements, as well as system and nonfunctional requirements. What a user can do dictates quite a bit of what the actual app should do, although the implementation details of the functionality is abstracted away from the user. Creating a UML Use-Case Diagram provided a clear declaration of the different functions that the various users should be able to perform. Using this diagram, UML Activity and Sequence Diagrams were quite easy to construct and showed the inner processes of different use cases and the sequence of events that must take place. Finally, I created a UML Class Diagram to show the classes (objects) that would be needed for implementation of these designs, including their associated attributes and operations and relationships amongst these classes.

When it comes to designing software, essentially the first step to take is understanding the issue(s) and desired solution(s) that motivated the need for a new system or app. This may require use or understanding of the existing system, including the business model (and industry), as well as interviewing users, looking at business forms, and gathering other pieces of relevant information. After gathering requirements for the system, I would create documents such as the Baseline Project Plan and Business Requirements Document (BRD). These can be shown to clients and stakeholders to gain approval of the project and the direction it is intended to take. Upon approval, visual aids like UML Use-Case, Activity, Sequence, and Class Diagrams would be created. Then, using this accumulated data, I would create a System Design Document (SDD) to organize and convey these system details to coders, testers, stakeholders, and associated managers. The SDD would also be an invitation for feedback and talking through any technical implementation concerns (for example) and resolving any other issues of concerns.

After all vested parties have signed off on the design and implementation details presented in an updated SDD, the implementation of the system to a functioning program may begin. Once the system is coded and thoroughly tested and documentation (including training and support) is complete, the system may be installed into a live production environment, initially with a partial install (preferred) before installing to the entire production environment. After this, it's important to fix any bugs that may not have been caught before installation. So, I would plan to collect maintenance requests from system users, auditors, data center and network management, and data analysts (if available). These requests could then be analyzed, along with risk/cost analysis if necessary, and then transformed into actual system changes, upon approval via a Change Request, such as a System Service Request (SSR). The system code could then be updated (coded and tested), new documentation could be created, and the new system installed.

These are basic steps of the System (or Software) Development Lifecycle (SDLC), which includes iterating through the first phases (Planning, Analysis, Design, Implementation) when executing the final phase (Maintenance), which is in line with modern, iterative developmental frameworks like Agile, typically leading to greater business value at a higher frequency of return! And this is exactly how I would implement future analysis and design, dependent on the specific scenario.

NOTE: I have included the BRD and SDD (0.1) for the DriverPass system in this repository.

Thanks!