# System Level Design

<Provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.>

Per the Software Requirements Specification (SRS), the Handicapped Driver design is required to “provide user/driver interaction via web browsers on both Android and iOS smart devices and Windows and OSx operating systems” (SRS 1.1); therefore, a presentation layer (GUI) was designed as an independent section and partitioned accordingly. Additionally, the system must store information related to parking lots/spaces, registered drivers and ongoing reservations made by drivers for the spaces. Therefore, an independent data layer (database) was also designed and partitioned. The core of the system remains as its own layer and contains both business logic and components to interact with the aforementioned presentation and data layers. The core also contains adapters as interfaces that implement features facilitated by external systems such as communication (SMS and Email), road navigation (Google Maps), and the scheduled “cleanup” of internal data. Overall, the Handicapped Driver System is architected as a multi‑tier software solution within a web‑based client‑server paradigm.

## Hierarchy Chart Of Components

<Describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its components/subsystems (identifying each top-level component/subsystem and the roles/responsibilities assigned to it). Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). >



Figure 4.1 – Package Diagram

## Sequence Diagrams And Descriptions

<Describe the interaction behavior between classes. The typical sequence diagram will depict some or all of the behavior described in a use case. Therefore, sequence diagrams serve as an intermediary between use cases and classes. As a result, classes can be traced back to the requirements. >

Kghfkg



Figure 4.2a – User/Driver Login



Figure 4.2b – Remind Driver



Figure 4.2c – Make Reservation



Figure 4.2d – Use Reservation (Park, Leave or Cancel)