# Get up and running directly

The first thing that’s required is a database with the name “cars”. Initialize this mysql database before deploying the program.

The program is initialized with a single staff entity. The credentials of this entity are

firstName: "Linda"

lastName: "Chua"

userName: “manager"

password: "password"

You can therefore login to the system in the same way as described in the document.

# System structure

We will make a system based on the following components:

* Database
* Core backend
* SOAP web service (web module)
* Clinic admin terminal (client)
* Self-service kiosk (client)
* AMS Client (“Consuming” web client)

## 

## Core backend

The core backend is comprised of the following session beans

### Stateless session beans

* AppointmentSessionBean

This session bean controls the functionality associated with making an appointment. It does not directly print things to console, since this will be done a bit differently for the different interfaces. It does however provide a set of methods for viewing and managing appointments.

* DoctorSessionBean

Handles the high level methods associated with the doctors. Things like requesting for leave etc.

* + We use java.sql.Date for the CRUD operations. In some parts of the backend, where manipulation of dates is required, we use other packages.
* PatientSessionBean

Handles CRUD operations for the patient entities. Also provides business methods to be accessed via the interfaces

* StaffSessionBean

Handles CRUD operations for the staff entities. Also provides business methods to be accessed via the interfaces

* AMSWebService
* Handles the SOAP for AMSClient

### Singleton session beans

* DataInitializationSessionBean

Handles initialization of the data. Basically the only entry that is put into the database is one staff member. The rest of the entities can be inputted via the system itself. The predefined entry has the following attributes:

firstName: "Linda"

lastName: "Chua"

userName: “manager"

password: "password"

* QueueSessionBean

Manages the queue numbers that patients acquire upon entering a consultation session.

## Clinic admin terminal

The clinic admin terminal is one of two Java Enterprise Application Clients associated with the system. The modules constituting the client are

* Main

Simply calls MainApp

* MainApp

Handles login and registration and therefore makes use of the StaffSessionBean and the PatietSessionBean. Redirects to The proceeding 3 modules for specific operations. Stores the id of the staff using it.

* AdministrationModule

Handles administration operations, therefore communicates with the AdministrationSessionBean. The module stores the staff id currently modifying and provides it to the Session Bean.

* + Patient management
  + Doctor management
  + Staff management
* AppointmentModule

Contains the staff interface associated with performing the Appointment operations.

* + View patient appointments
  + add appointment
  + cancel appointment
* RegistrationModule

Contains the staff interface associated with performing the Registration operations.

* + Register new patient
  + Register walk in consultation
  + Register consultation by appointment
* TimeFiltrator

Maintains a set of helper methods for the other modules in the clinicAdminTerminal. This reduces clutter in the other modules.

## SelfServiceKiosk

The clinic admin terminal is one of two Java Enterprise Application Clients associated with the system. The modules constituting the client are

* MainKiosk

Simply calls MainAppKiosk

* MainAppKiosk

Handles login and registration and therefore makes use of the PatientSessionBean. Redirects to The proceeding 1 module for specific operations. Stores the id of the patient using it. Handles Registration of patients and logging in for patient.

* + Register new patient
  + Login
* MainModuleKiosk

Contains the interface associated with performing the Kiosk operations. Handles Kiosk operations. The module stores the patient id currently modifying and provides it to the respective Session Bean.

* + View patient appointments
  + Add appointment
  + cancel appointment
  + Register walk in consultation
  + Register consultation by appointment
* TimeFiltratorKiosk

Maintains a set of helper methods for the other modules in the SelfServiceKiosk. This reduces clutter in the other modules.

## AMSClient

The clinic admin terminal is one of two Java Enterprise Application Clients associated with the system. The modules constituting the client are

* AMSClient

Simply calls MainAppAMS

* MainAppAMS

Handles login and registration. Redirects to The proceeding 1 module for specific operations. Stores the id of the patient using it. Handles Registration of patients and logging in for patient.

* + Register new patient
  + Login
* PatientMenu

Contains the interface associated with performing the AMS operations. Handles AMS operations. The module stores the patient id currently modifying and provides it to the respective Session Bean.

* + View patient appointments
  + Add appointment
  + cancel appointment

## Database

The name of the database is ”cars”. It should be all lowercase and needs to be instantiated before the software is deployed.

there are 3 Database tables provided in the instruction document (page 3). These are from the different types of personnel that can interact with the system.

* StaffEntity
  + Long: staffId
  + String: firstName
  + String: lastName
  + String: userName
  + String: password
* DoctorEntity
  + Long: doctorId
  + String: firstName
  + String: lastName
  + String: registration
  + String: qualifications
* PatientEntity
  + Long: patientId
  + String: identityNumber (must be unique)
  + String: firstName
  + String: lastName
  + Enum: gender (can you store enum in database)
  + Integer: age
  + String: phone (could have picked integer but string is more flexible)
  + String: address
  + String: password

Other tables that we chose to include are:

* AppointmentEntity
  + Long: appointmentId
  + Long: doctorId (primary key in doctor table)
  + Long: patientIdentityNumber (unique identifier in patient table)
  + Date: date
  + Time: time
* DotorsLeaveEntity

Together the doctorId and the Date could serve as a primary key. This is because every combination of them must be unique in the table. However, we chose to make use of doctorsLeaveId, a surrogate key that uniquely identifies each element.

* + Long: Long: doctorsLeaveId (surrogate key)
  + Long: doctorId (primary key of DoctorEntity)
  + Date: date