Software Engineering Practice and Experience
17027925 - CW1

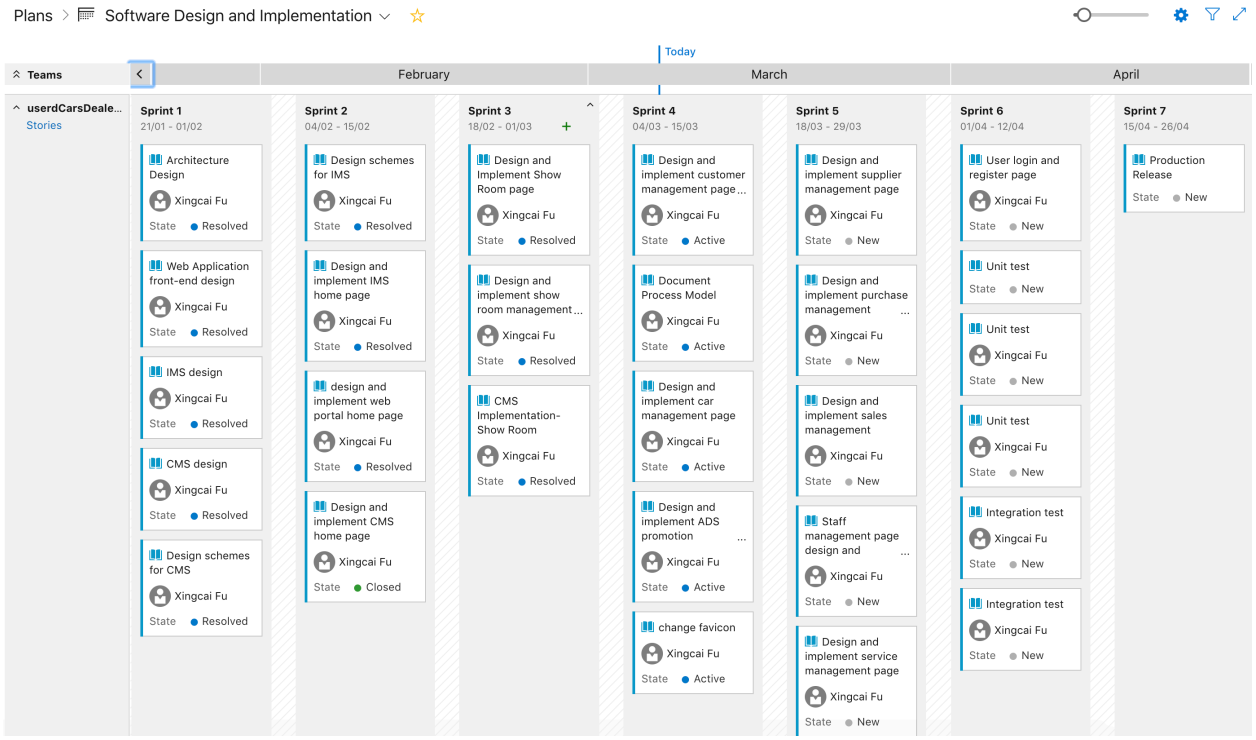
System Specification

Development Schedule

Sprint Plan

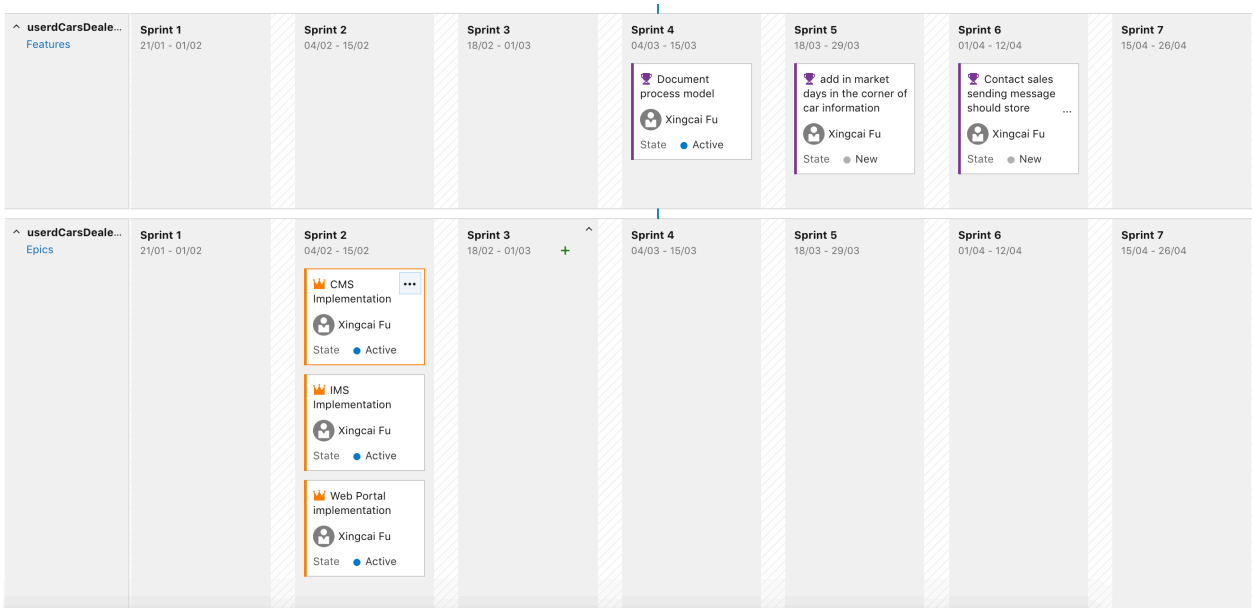
I'm using Microsoft Azure sprint plan extension to make my plan looks better. You can add multiple plans for your project. Here I added three plans which are user stories, features and EPICs.
There are seven sprints, and each sprint includes 2 weeks.

User Stories



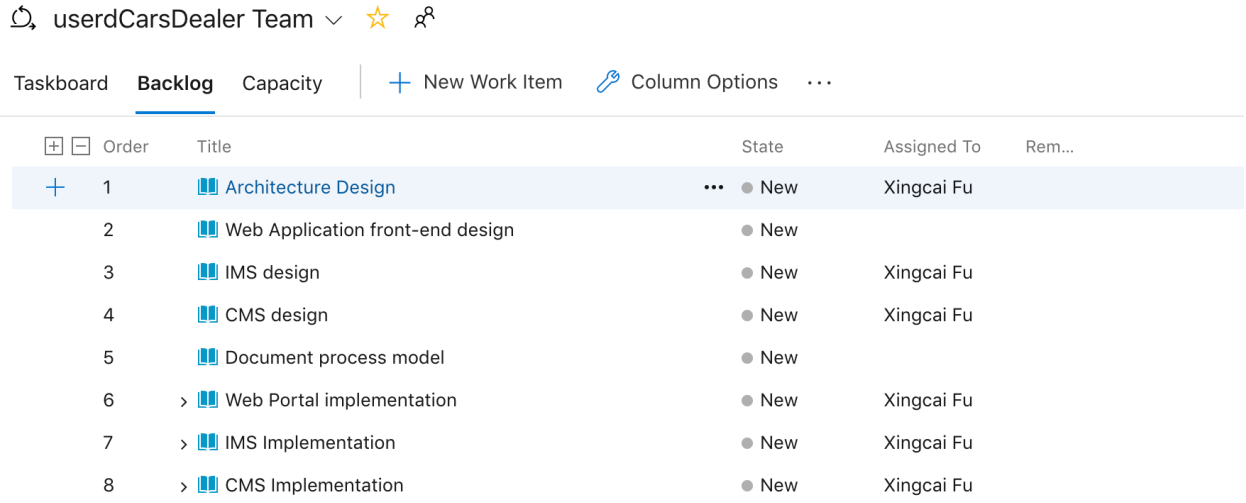
Since there are three modules in this application, I also setup three EPICs, which are Web Portal, IMS and CMS. All user stories or features are created from those three EPICs.

Features and EPICs



I set CMS, IMS and Web Portal as EPIC because Microsoft Azure cannot show tasks in the sprint plan. In Azure, I can only add task or bug under user story. I think the task in Azure is kind of items of sub-task in JIRA. It also cannot be added to sprint plans from sub-task in JIRA.

Architecture Design



During the first one or two sprints, I will spend most of my time on architecture design. It includes Web application front-end design, Information Management System design and Content Management System design.

Web Portal Implementation

+	6	Web Portal implementation	New	Xingcai Fu
		design and implement web portal home page	New	Xingcai Fu
		Design and Implement Show Room page	New	Xingcai Fu
		implement and design car detail information page	New	Xingcai Fu
		Design and Implement car search page	New	Xingcai Fu
		User login and register page	New	Xingcai Fu
		car show room only shows TWO videos	New	Xingcai Fu
		add carousel slide show for cars	New	Xingcai Fu
		change favicon	New	Xingcai Fu
		add in market days in the corner of car information	New	Xingcai Fu
		Contact sales sending message should store message...	New	Xingcai Fu
		Unit test	New	

CMS Implementation

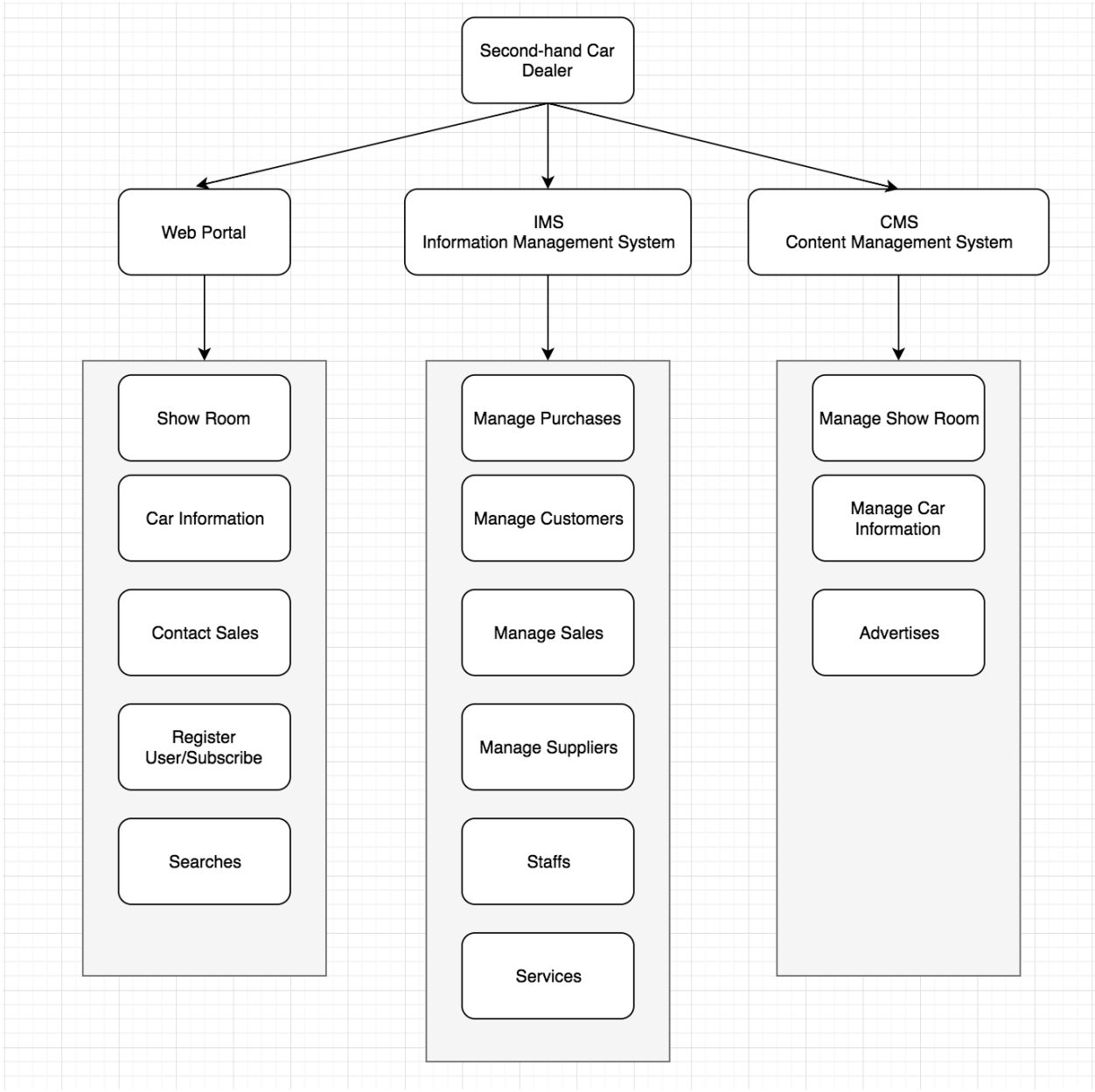
+	8	CMS Implementation	New	Xingcai Fu
		Design and implement CMS home page	New	Xingcai Fu
		Design and implement car management page	New	Xingcai Fu
		Design and implement show room management page	New	Xingcai Fu
		Design and implement ADS promotion management p...	New	Xingcai Fu
		Design schemes for CMS	New	Xingcai Fu
		Backend for CMS	New	Xingcai Fu
		Unit test	New	Xingcai Fu
		Integration test	New	Xingcai Fu

IMS Implementation

+	7	IMS Implementation	New	Xingcai Fu
		Design schemes for IMS	New	Xingcai Fu
		Design and implement IMS home page	New	Xingcai Fu
		Design and implement customer management page	New	Xingcai Fu
		Design and implement supplier management page	New	Xingcai Fu
		Design and implement purchase management	New	Xingcai Fu
		Design and implement sales management	New	Xingcai Fu
		Staff management page design and implementation	New	Xingcai Fu
		Design and implement service management page	New	Xingcai Fu
		Implement backend API endpoints for IMS	New	Xingcai Fu
		Unit test	New	Xingcai Fu
		Integration test	New	

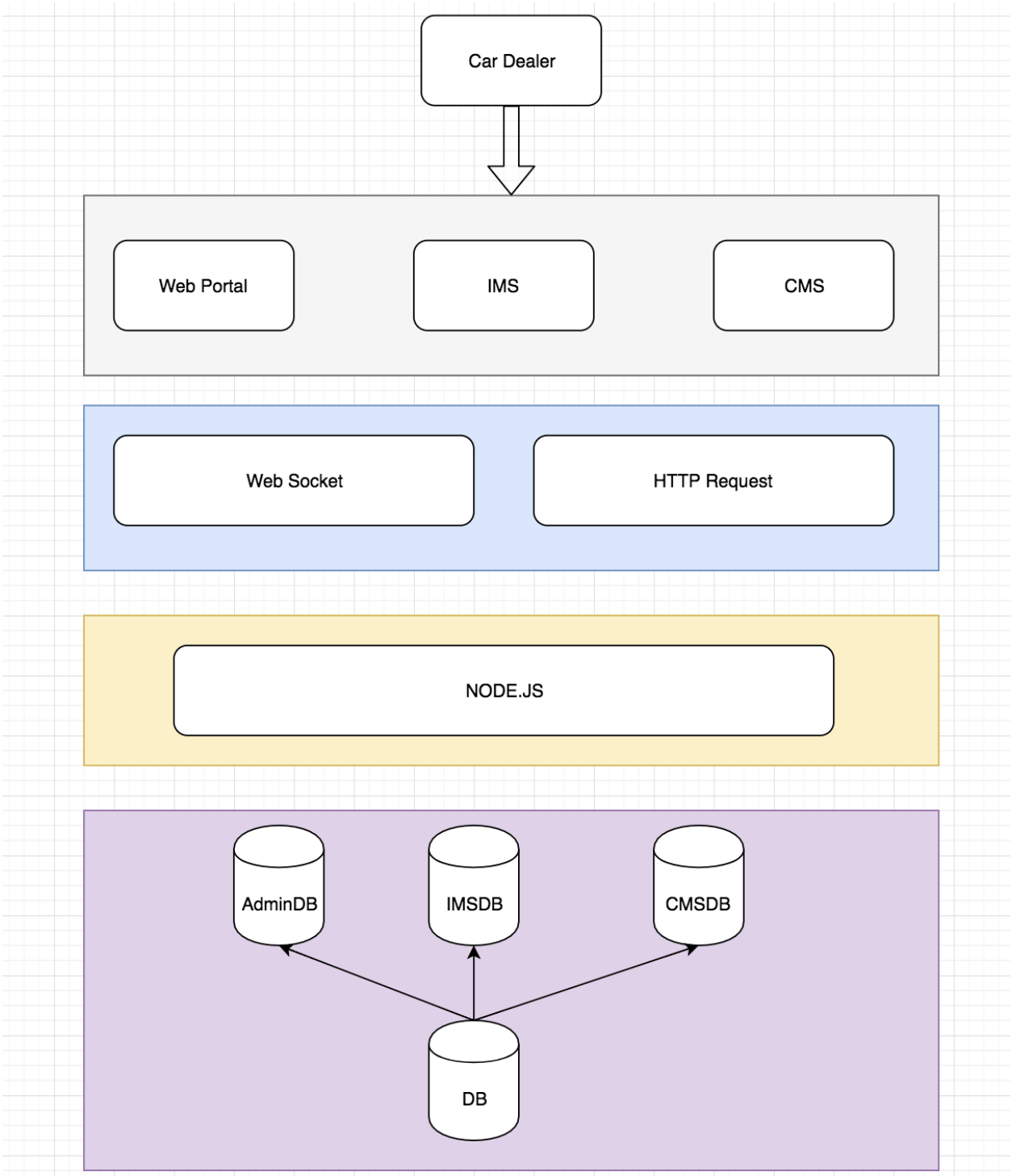
System Design

Architecture



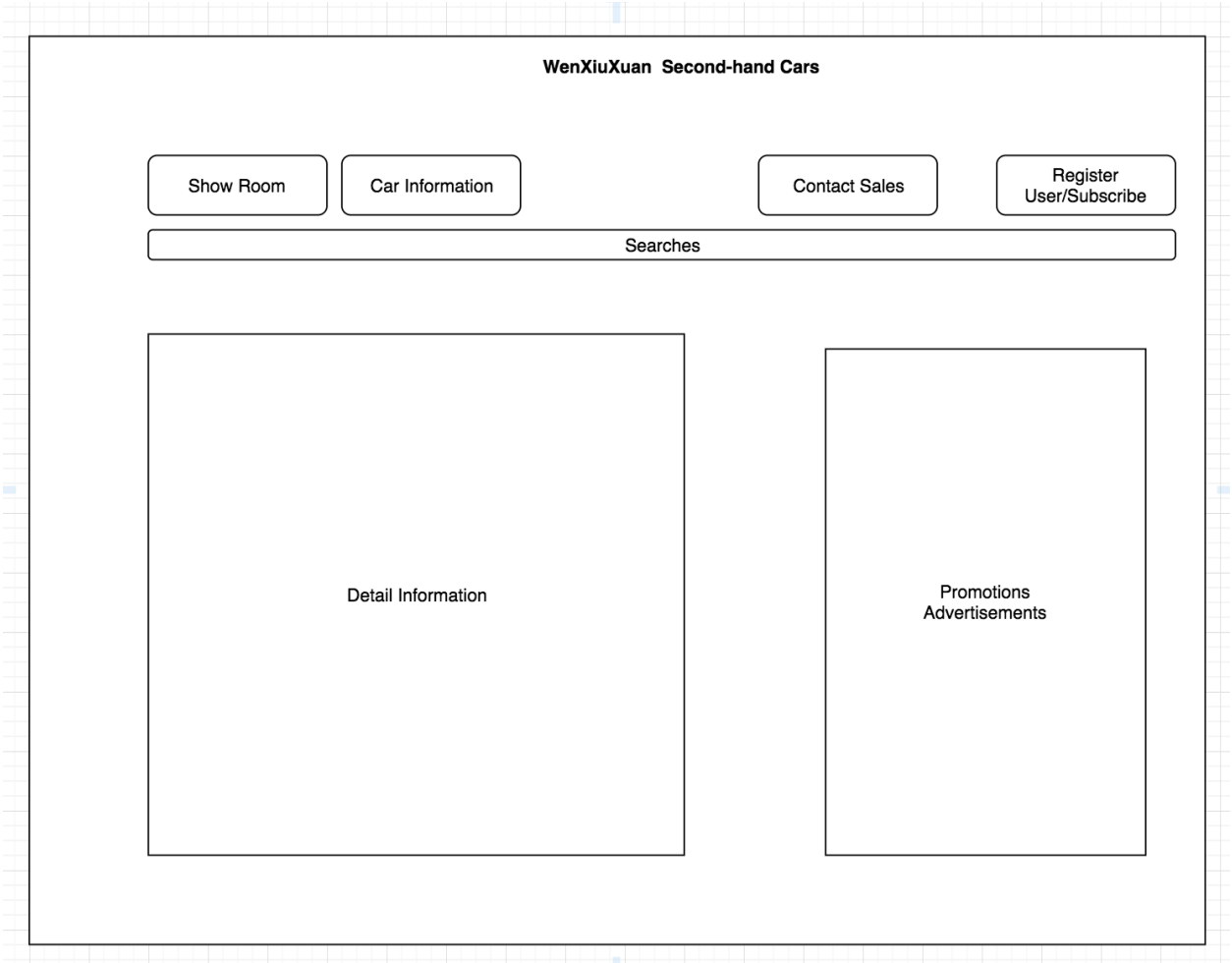
This system is using MEAN stack. The front-end framework uses Angular 7 with bootstrap, backend uses Node.js with express middleware, and database uses MongoDB.

The front-end sends HTTP or socket request to backend, after backend getting the request, Node.js will communicate with MongoDB and returns the result back to front-end.



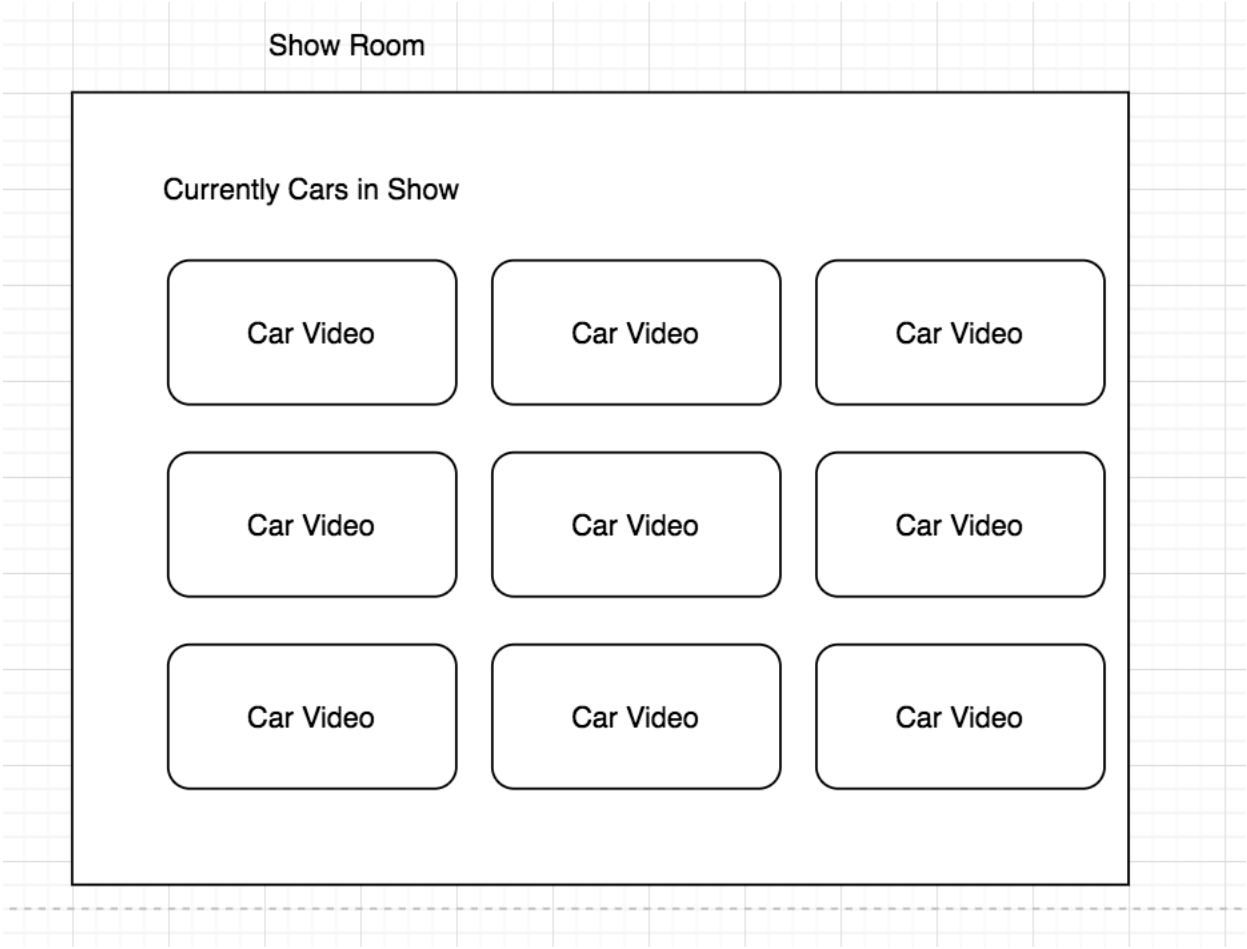
According to the design, all data for admin, IMS and CMS are stored in different DB schemes, but physically they are in the same location. The Admin DB contains user, client, staff and customer. IMSDB includes purchases, services, suppliers etc. CMSDB contains cars, car models, promotions and contacts.

Web



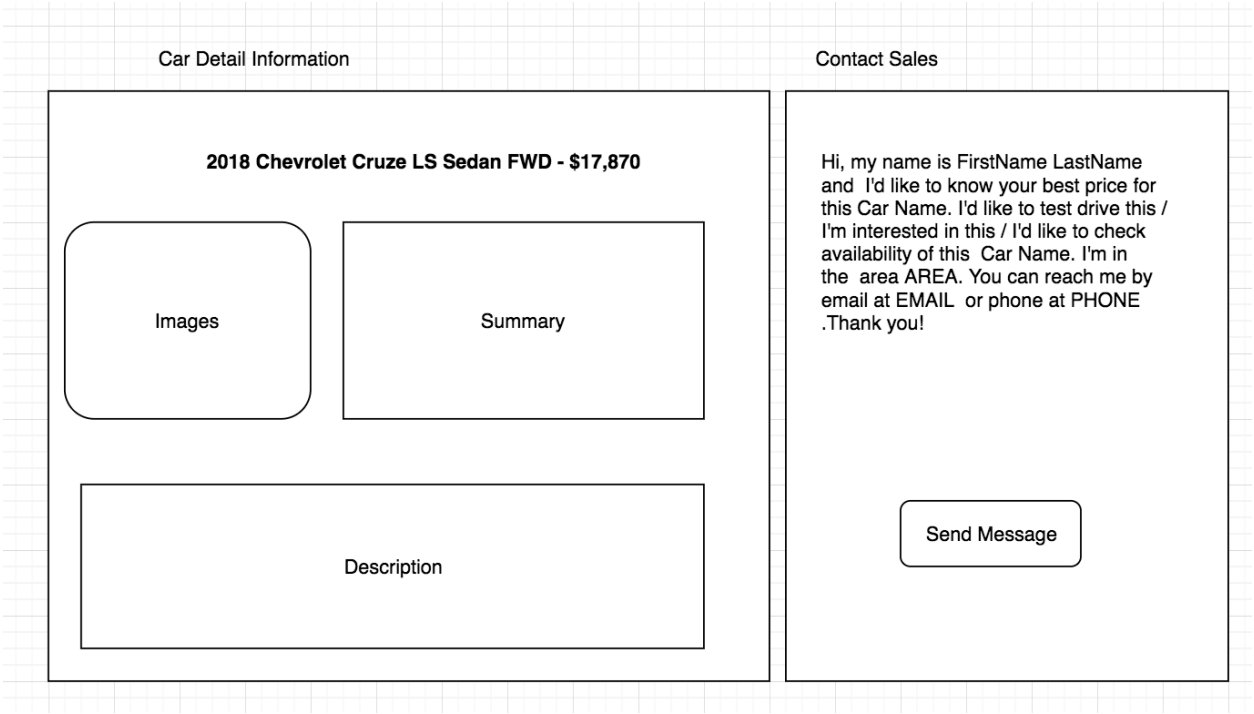
This is web UI home page scratch design. It contains navigation bar, car searching, car detail information, and promotions. The search button is designed as a bootstrap accordion. When clicking that search button, a panel with search conditions shows. When user clicking the “OK” button in search panel, all cars with the search criteria will show on the bottom. The right panel is designed to put advertisements, which will pull up from promotion component.

Show Room Page



The above chart shows the scratch design for car show rooms. All car videos are from YouTube. Currently I am using a third-party library to show YouTube videos in Angular. Maybe I will implement my own module to display the videos in the future sprint if time allowed.

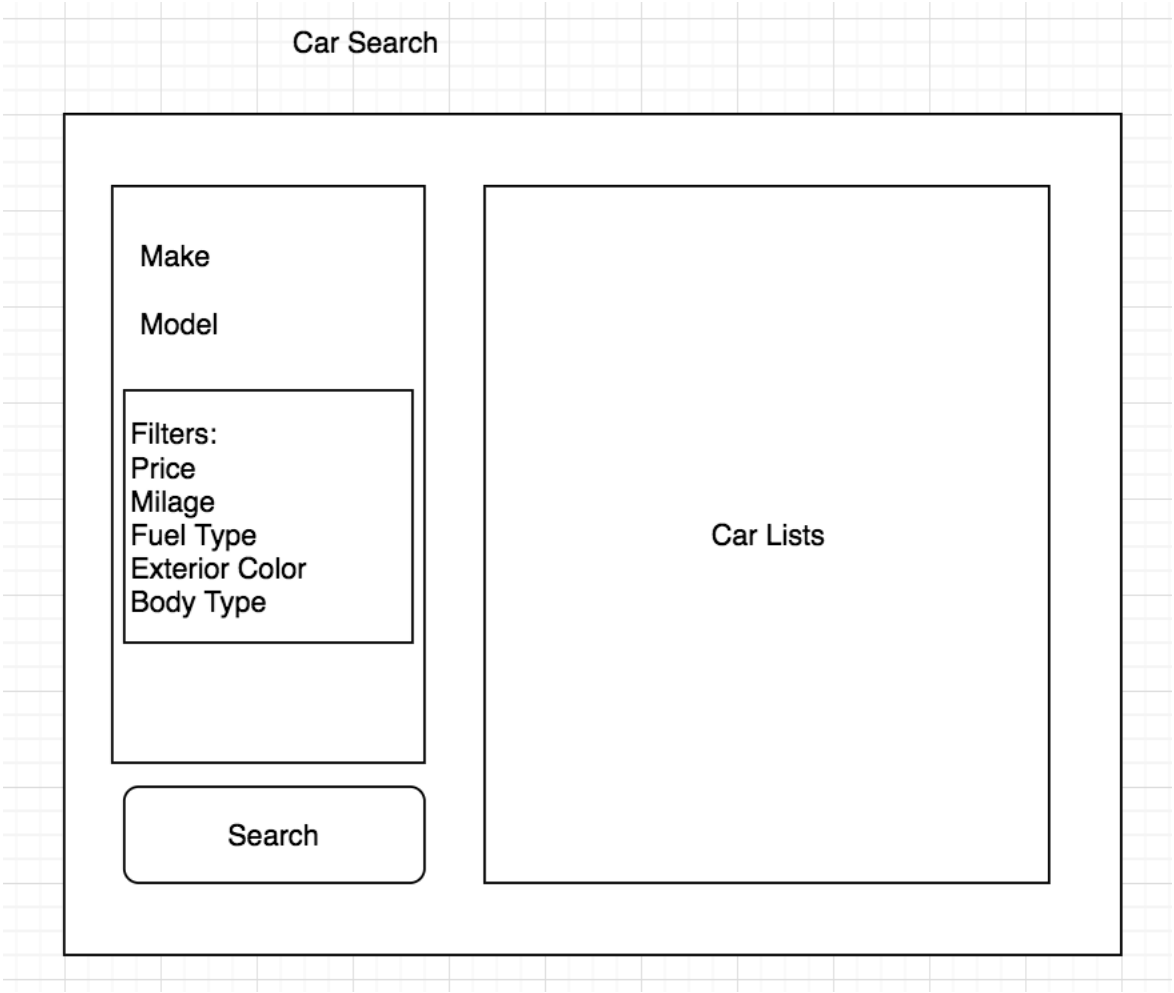
Car Detail Page



This page contains two parts. The left side shows the detailed car information, and the right side shows small window to connect with sales.

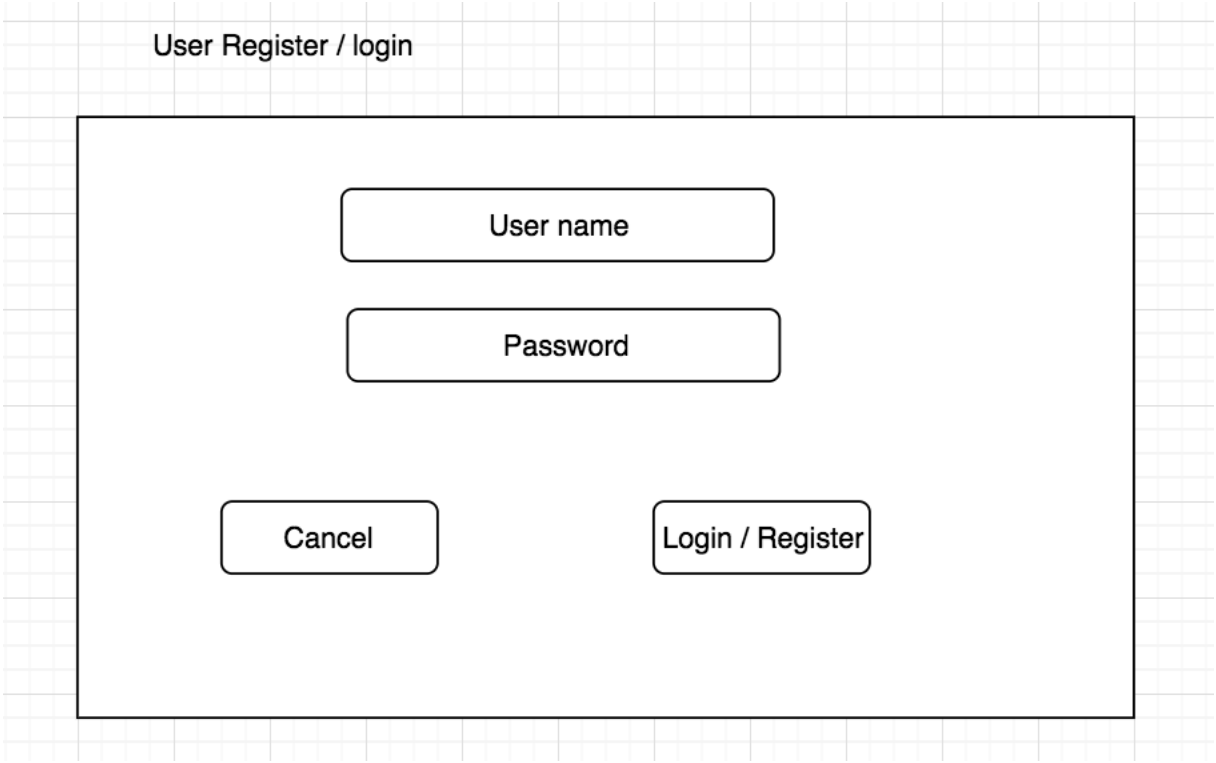
The right-side contact sales panel is supposed to show from the template. The user only need to fill a little of important information such as first name, last name and phone number or email address.

Car Search Page



Car search page shows the search criteria and results. When user selects vehicle make, mode and year etc., the result will be listed in the right side. If no car fits the query, it will show ‘No car returned’.

User Register and Login Page

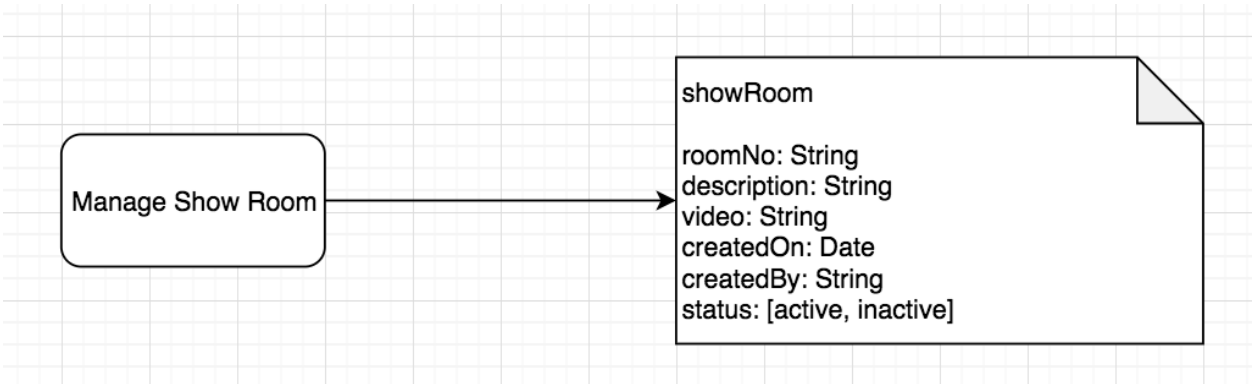


The user register / login page is a common page which could be used both for web, IMS and CMS. In the future JWT token will be applied to this page.

CMS

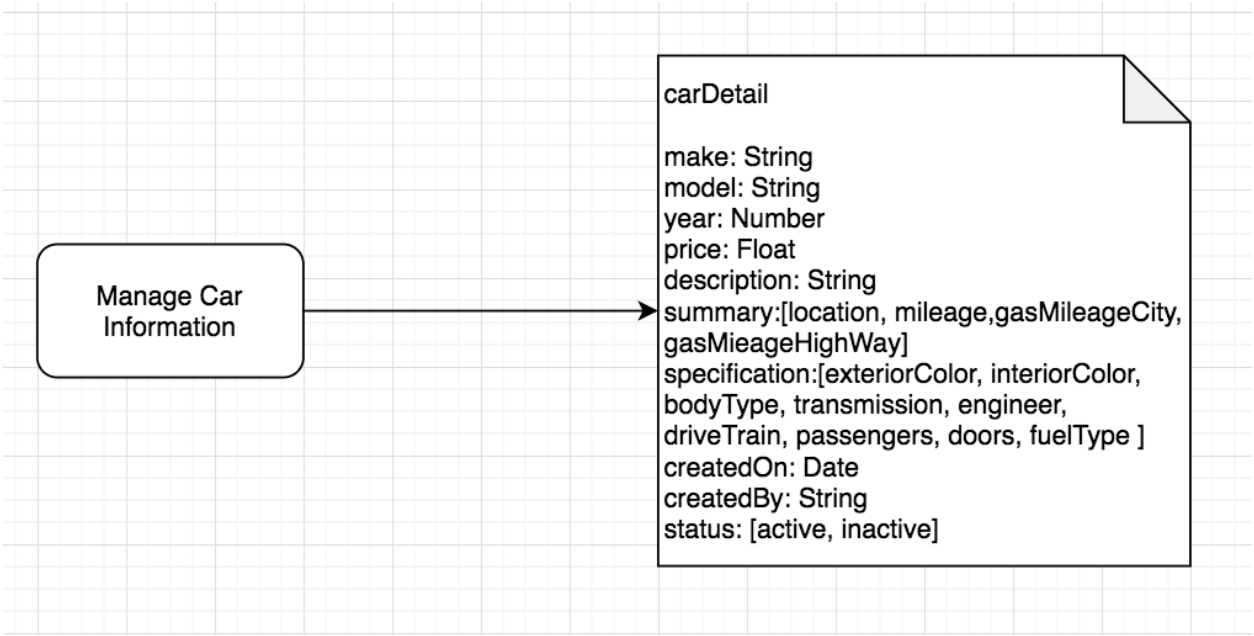
Scheme Design

1. Show Room



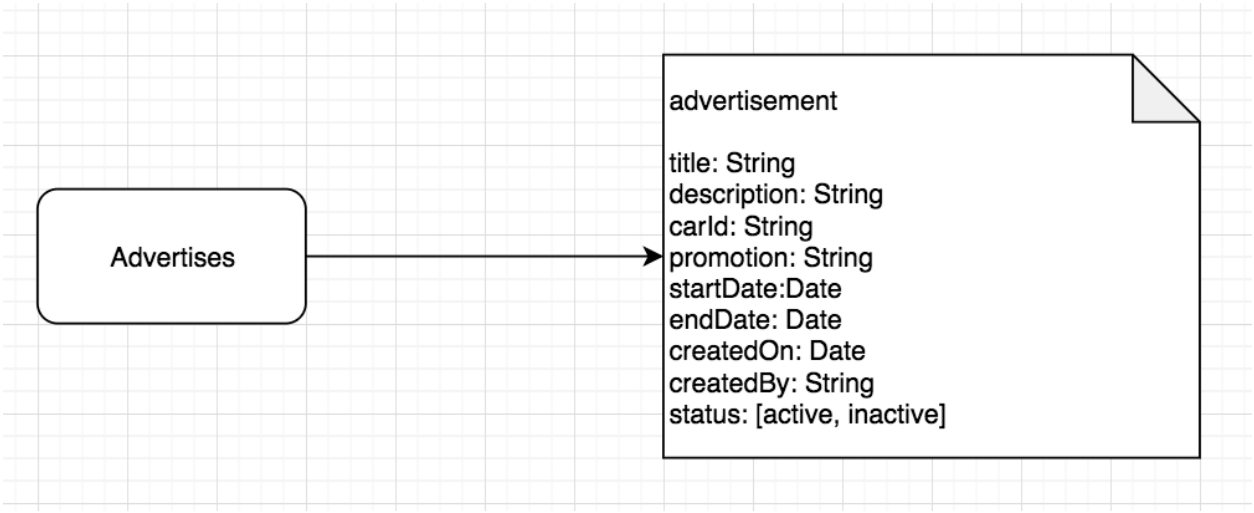
The video is an URL of YouTube. I don't store the whole video data in database.

2. Car Information

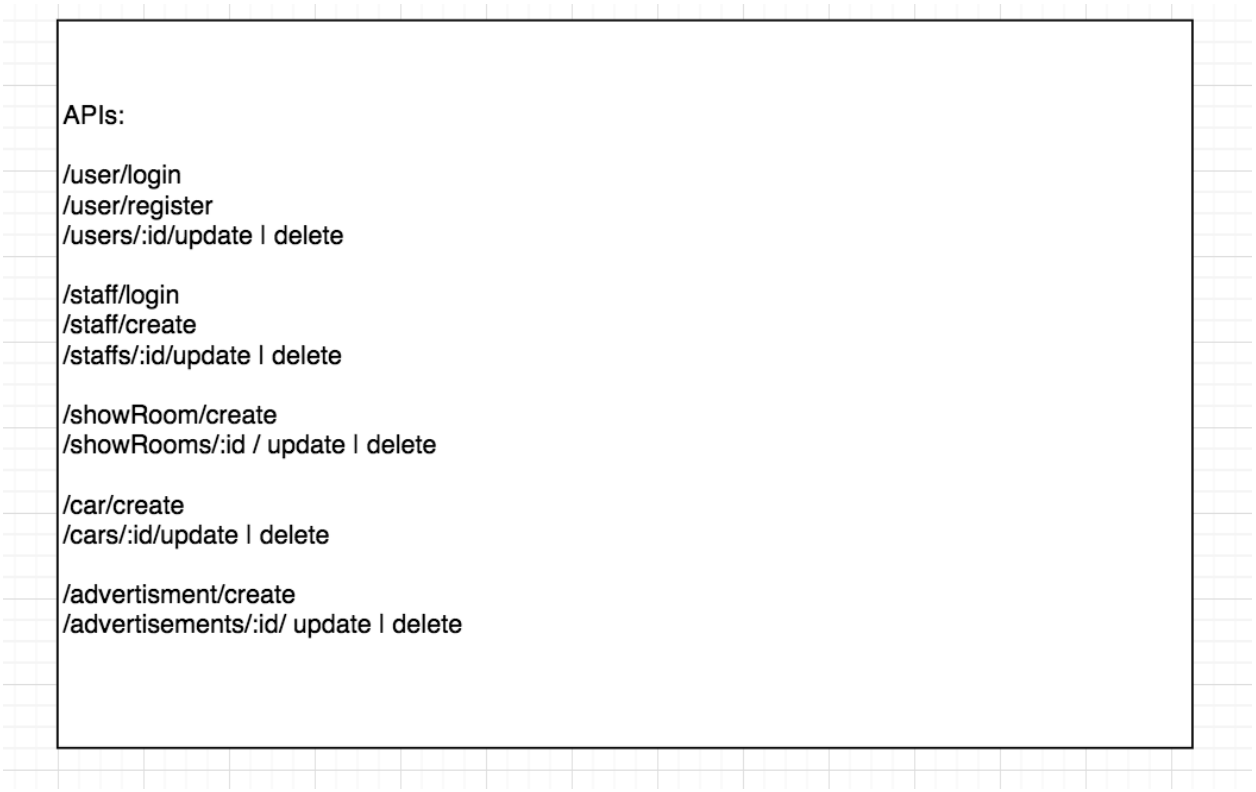


There's another scheme to keep car makes and models.

3. Promotion and Ads



API Router



There will be more routers need to implement.

Sales

- /sales/add
- /sales/remove
- /sales/update
- /sales/findAll
- /sales/find

Purchases

- / Purchases /add
- / Purchases /remove
- / Purchases /update
- / Purchases /findAll
- / Purchases /find

Suppliers

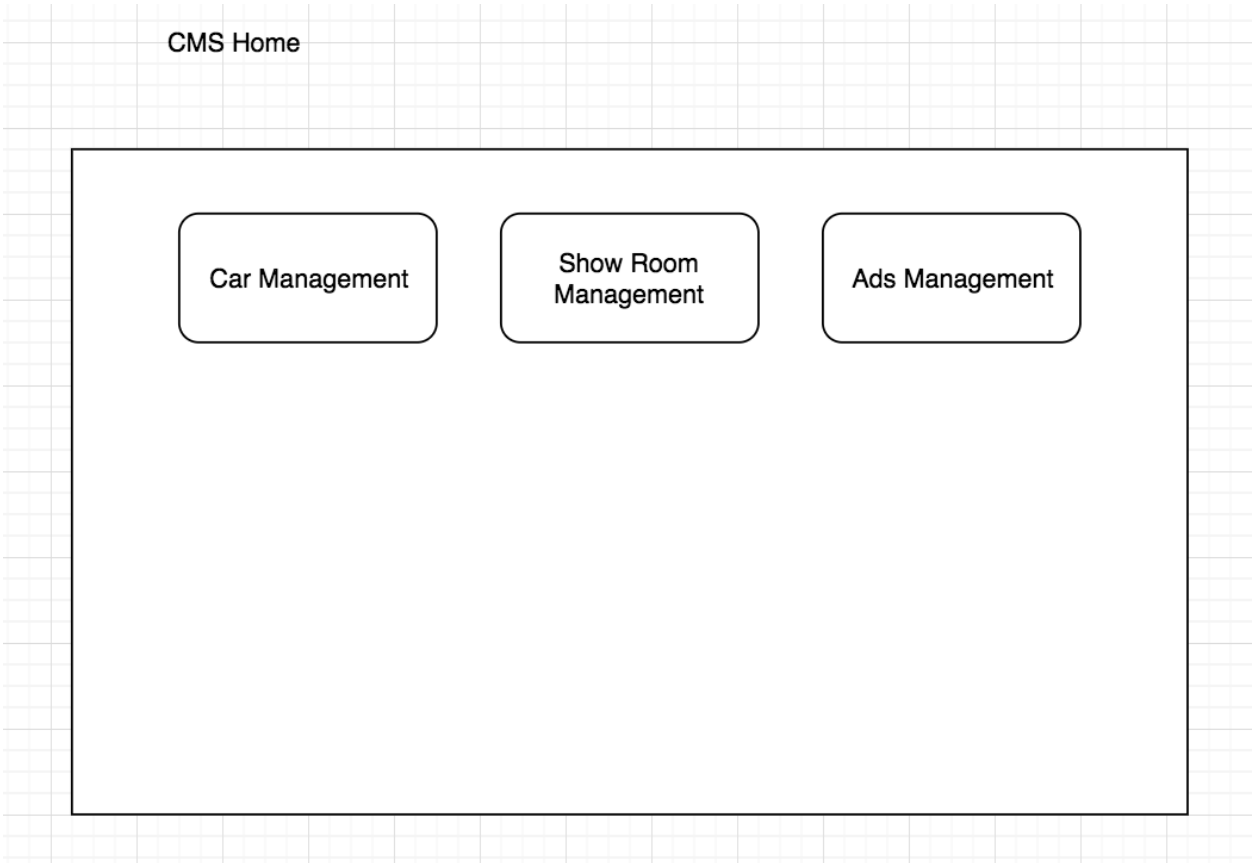
- / Suppliers /add

- / Suppliers /remove
- / Suppliers /update
- / Suppliers /findAll
- / Suppliers /find

Services

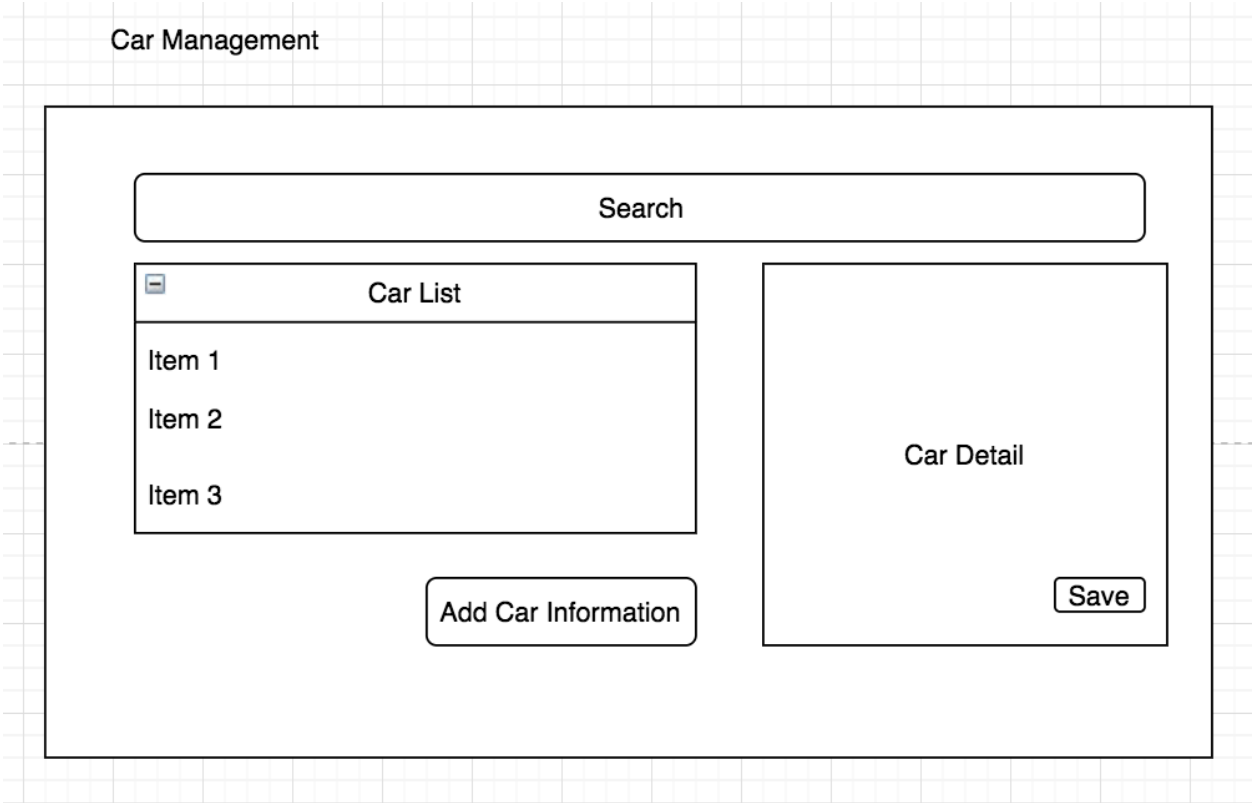
- / Services /add
- / Services /remove
- / Services /update
- / Services /findAll
- / Services /find

CMS Home



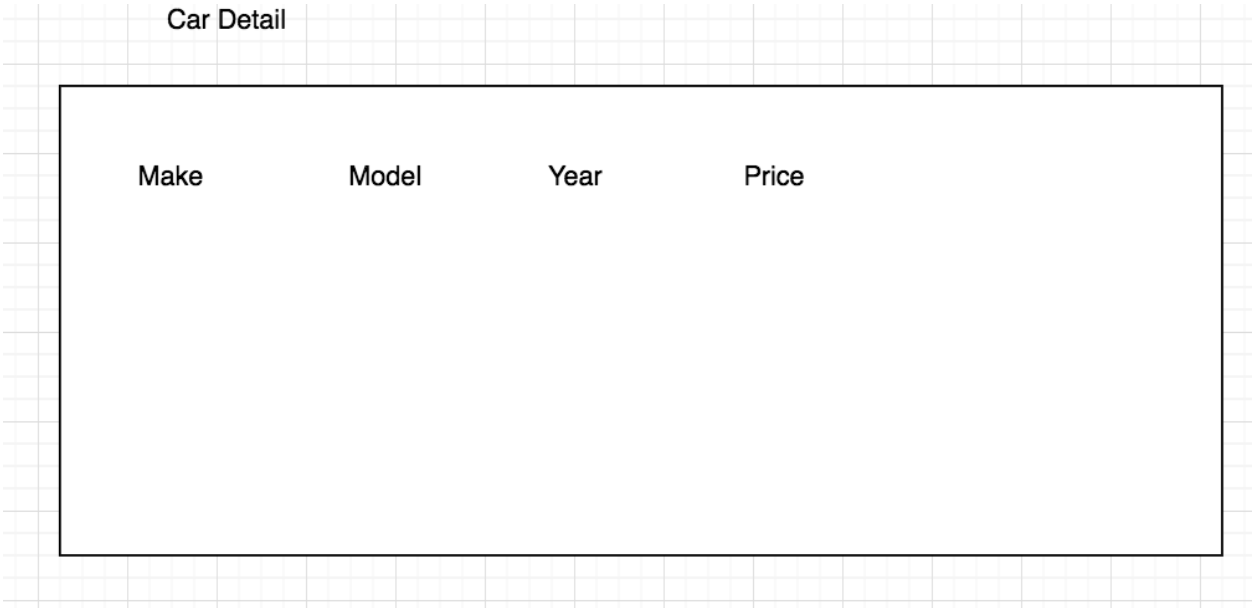
This CMS home page contains navigation bar of car management, show room management and promotion management.

Car Management

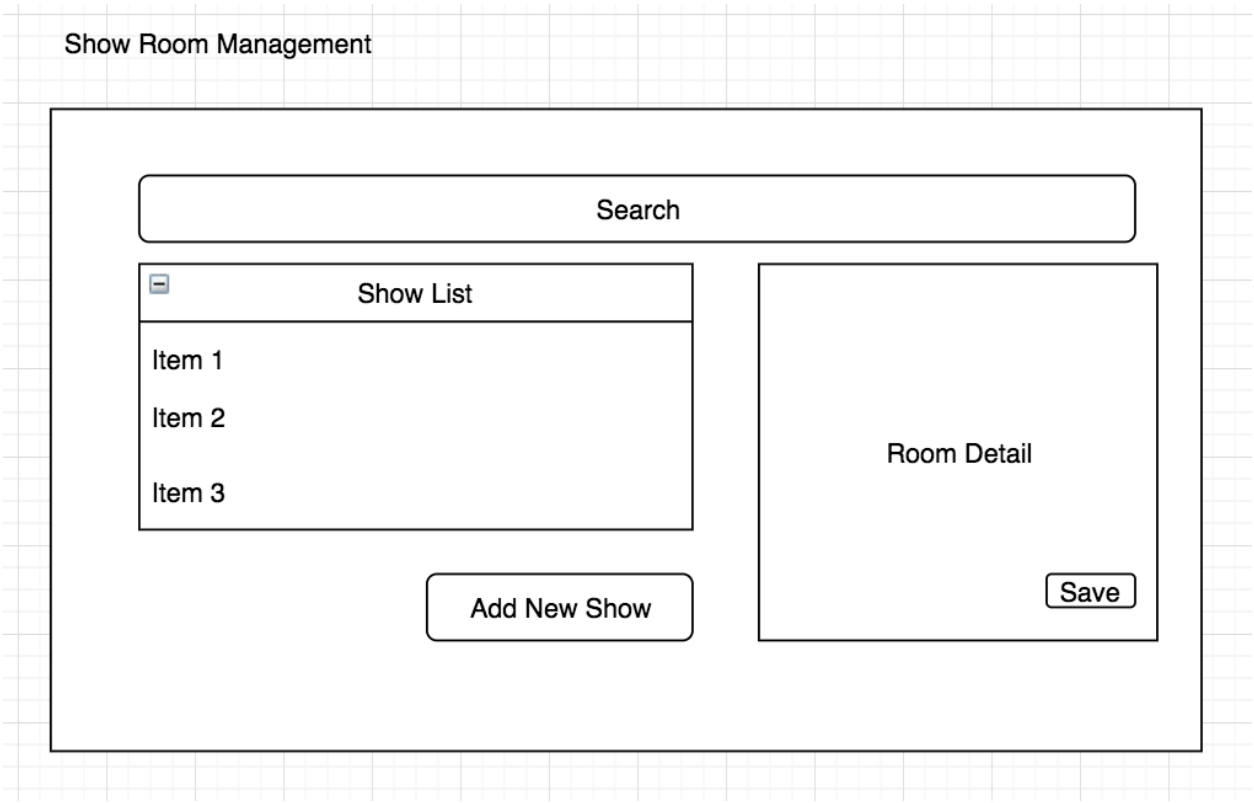


When clicking the item of left side, the car detail information will show in the right panel. User can add or update the car information and store to database. The user also can upload images for selected cars.

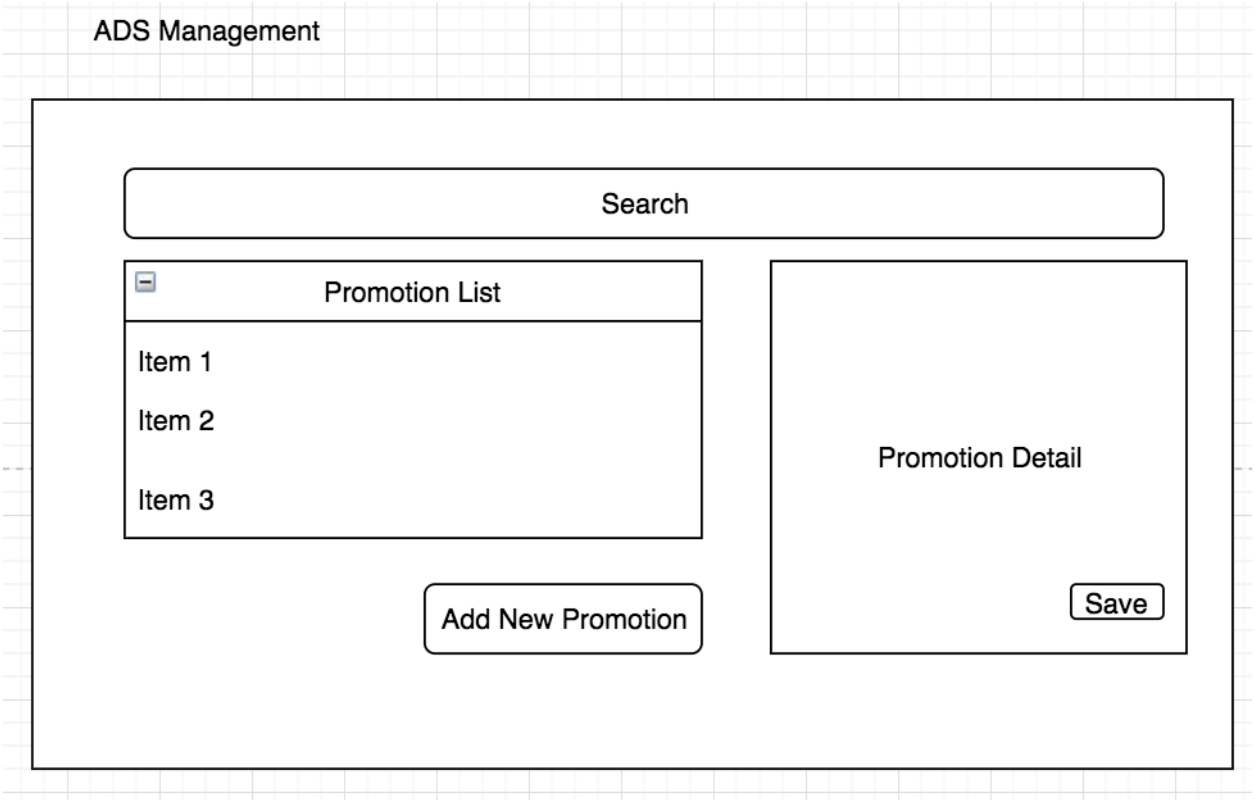
Car Details



Show Room Management



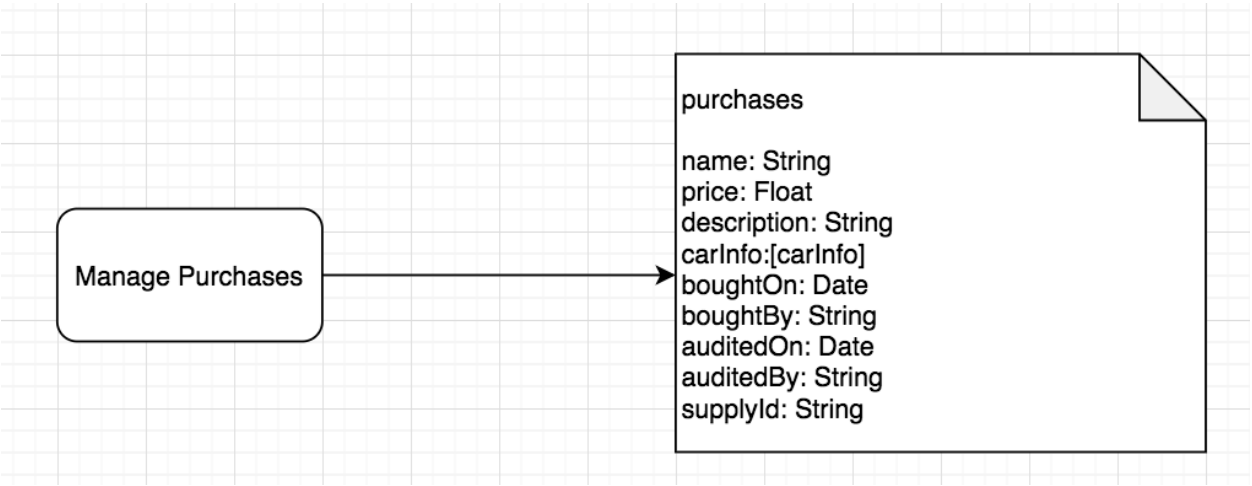
Promotion and Ads Management



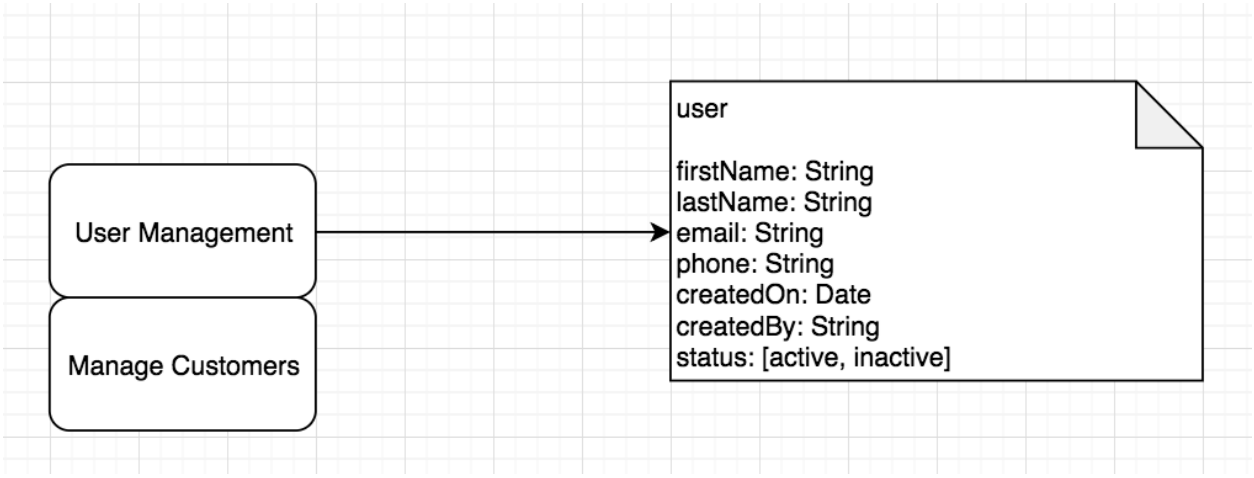
IMS

Scheme Design

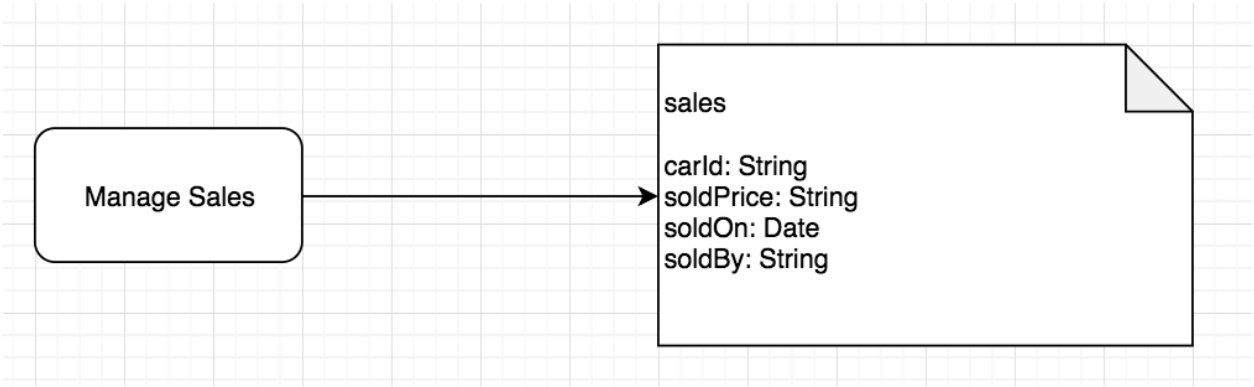
1. Purchase Management



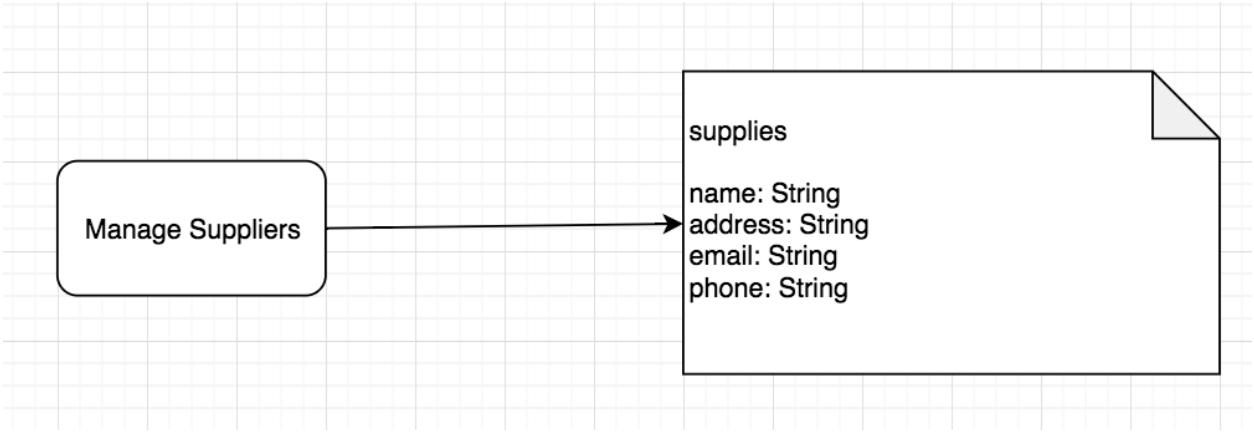
2. User Management



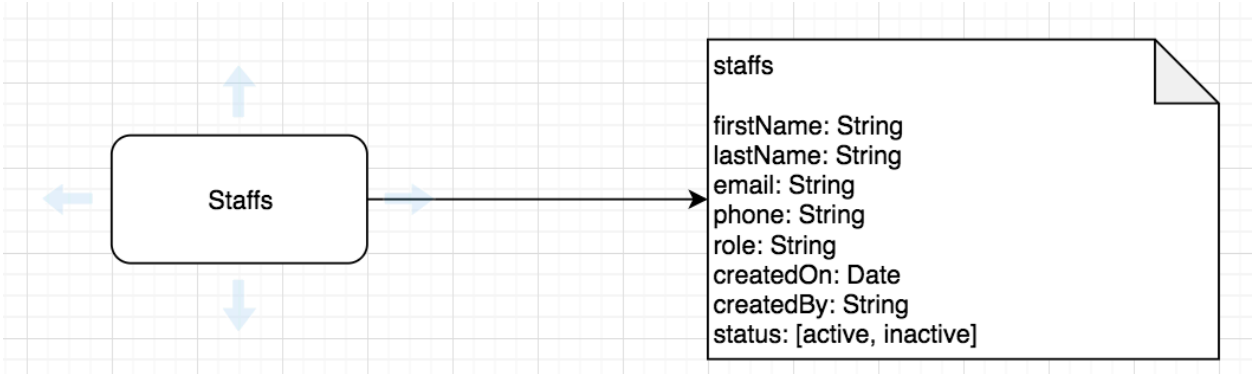
3. Sales Management



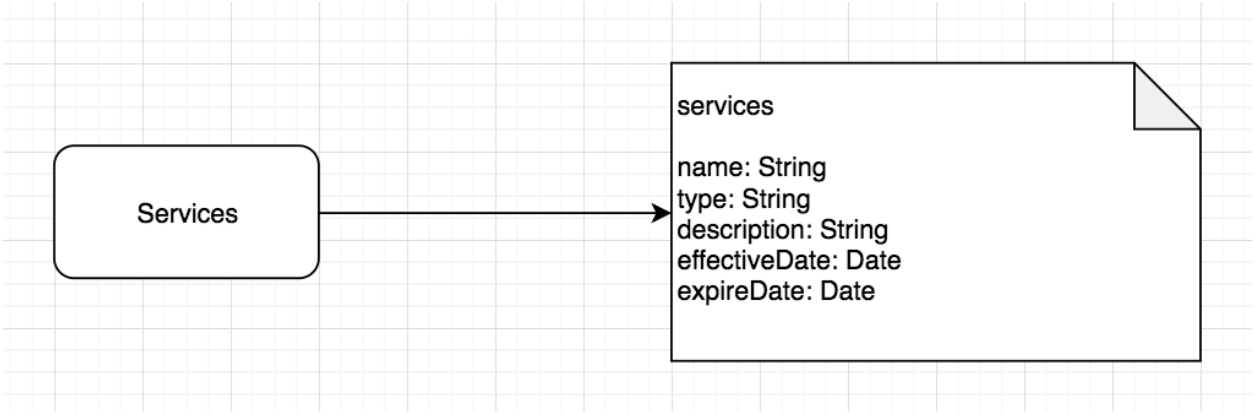
4. Supplier Management



5. Staff Management

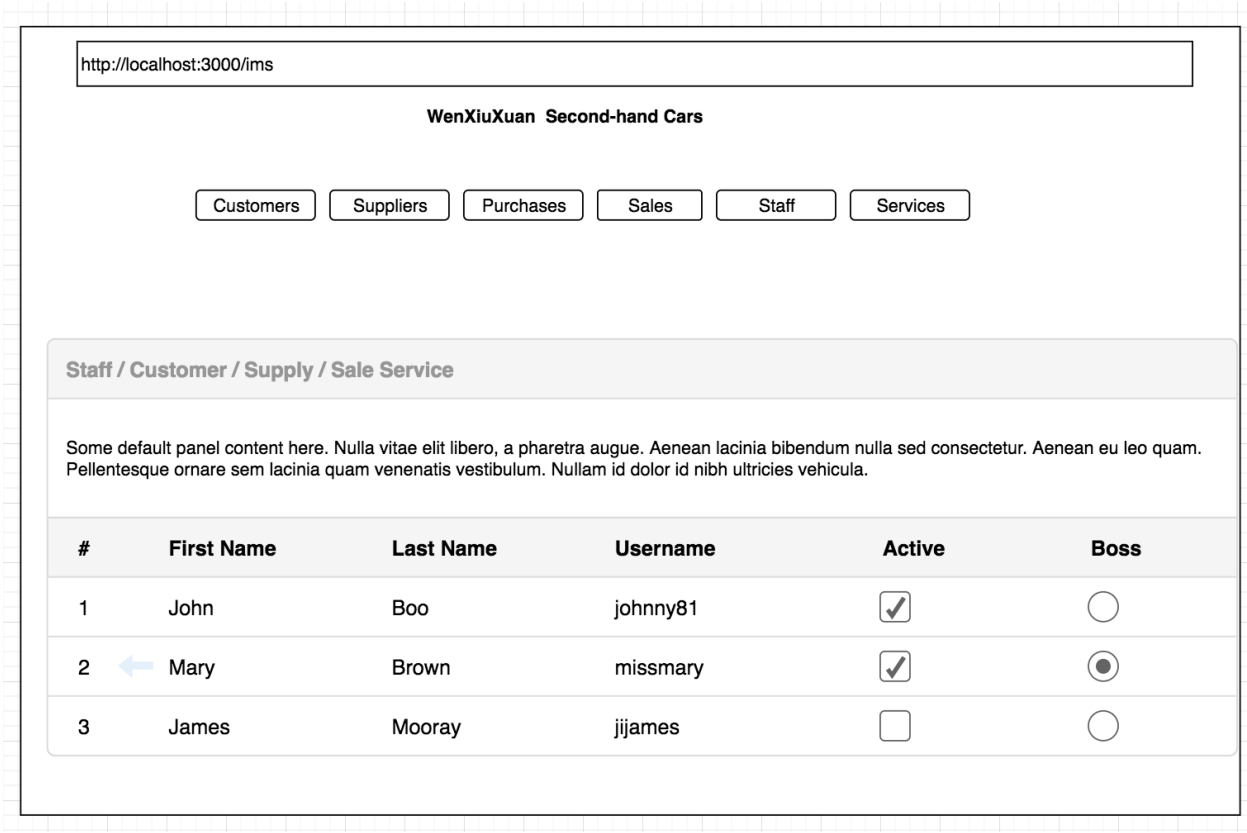


6. Service Management

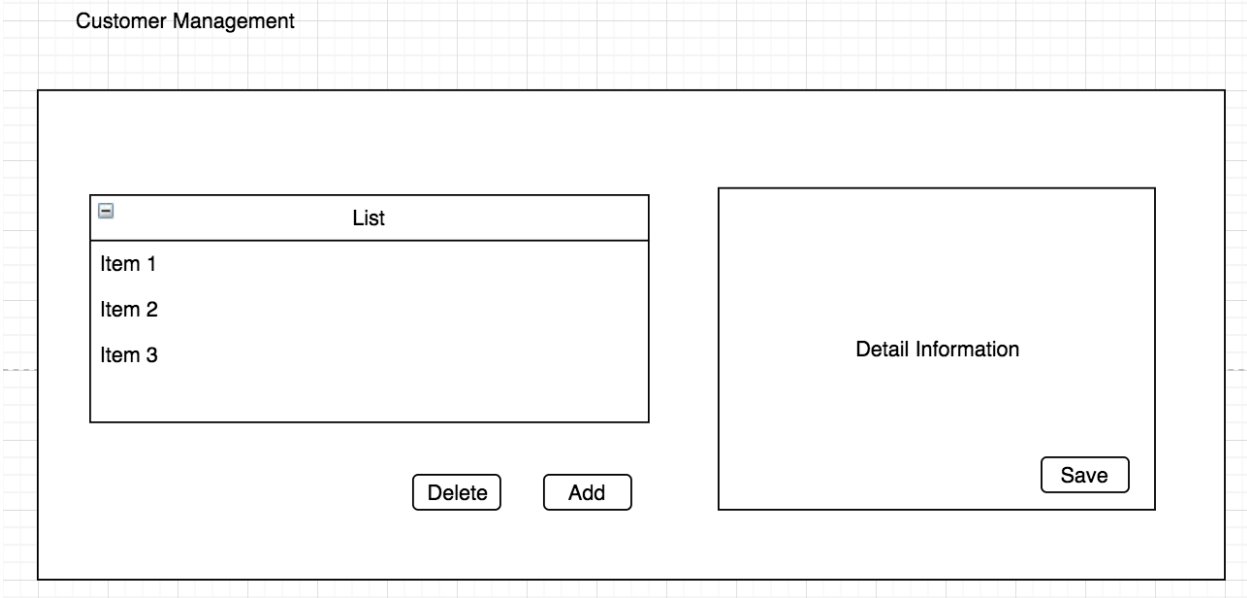


UI Design

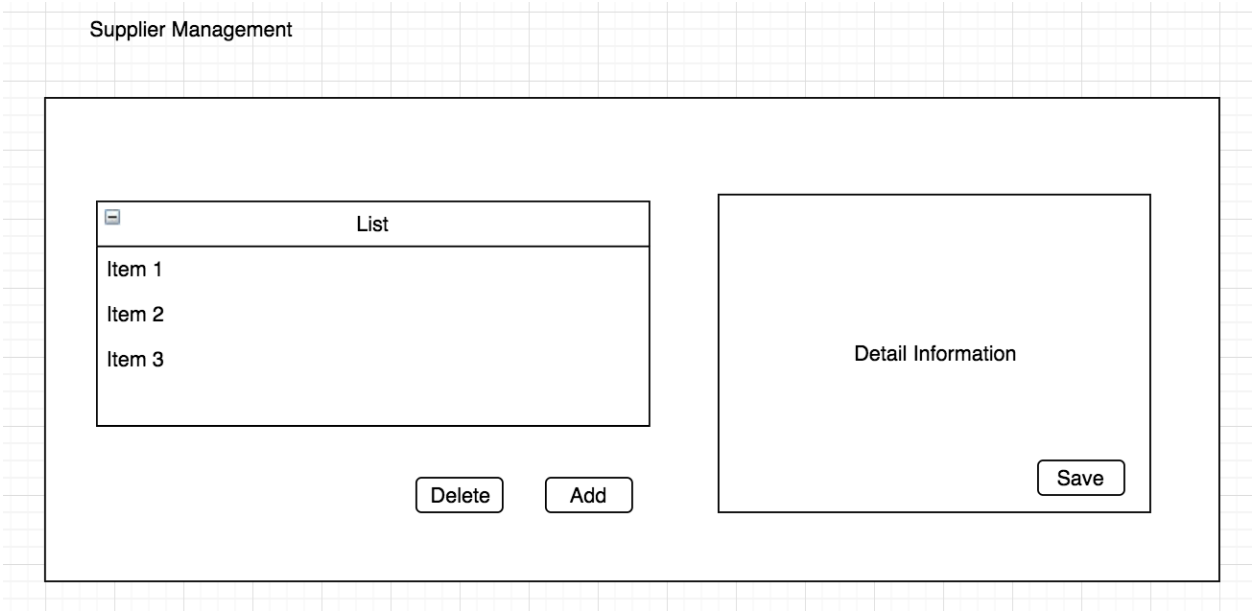
Home Page



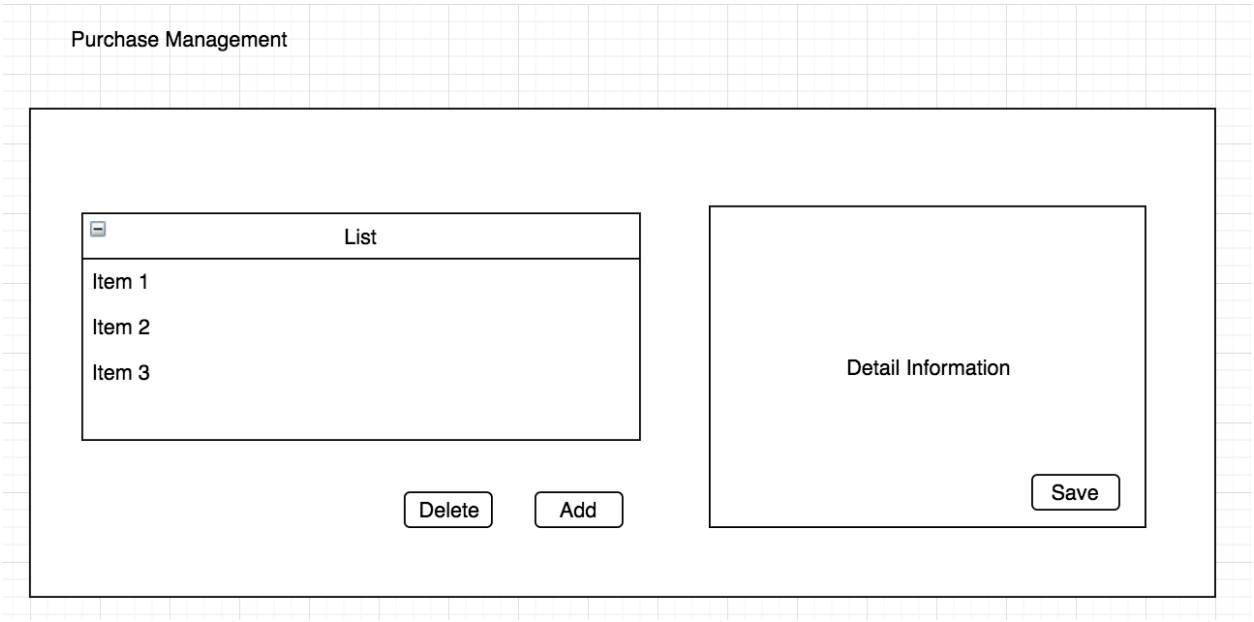
Customer Management



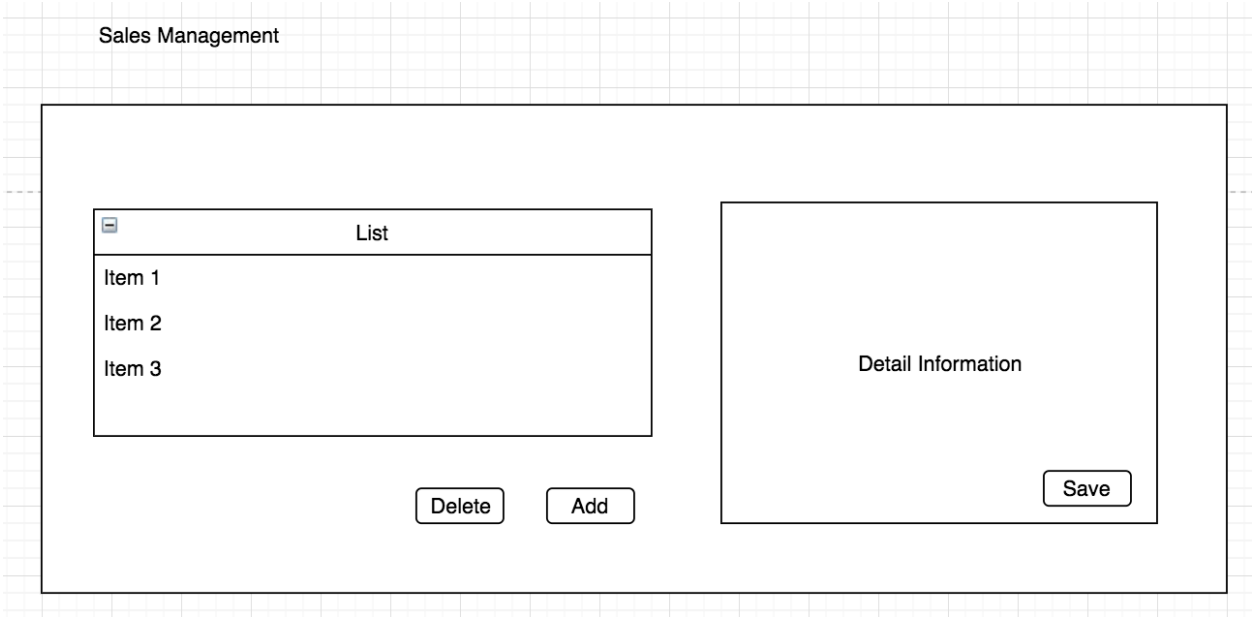
Supplier Management



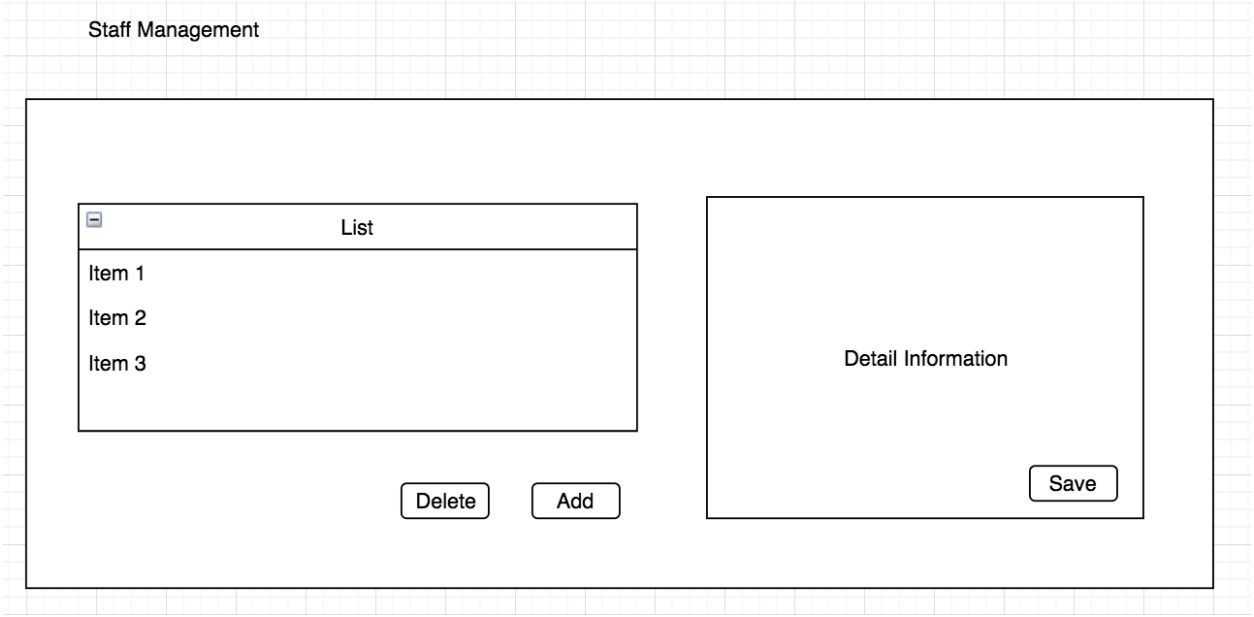
Purchase Management



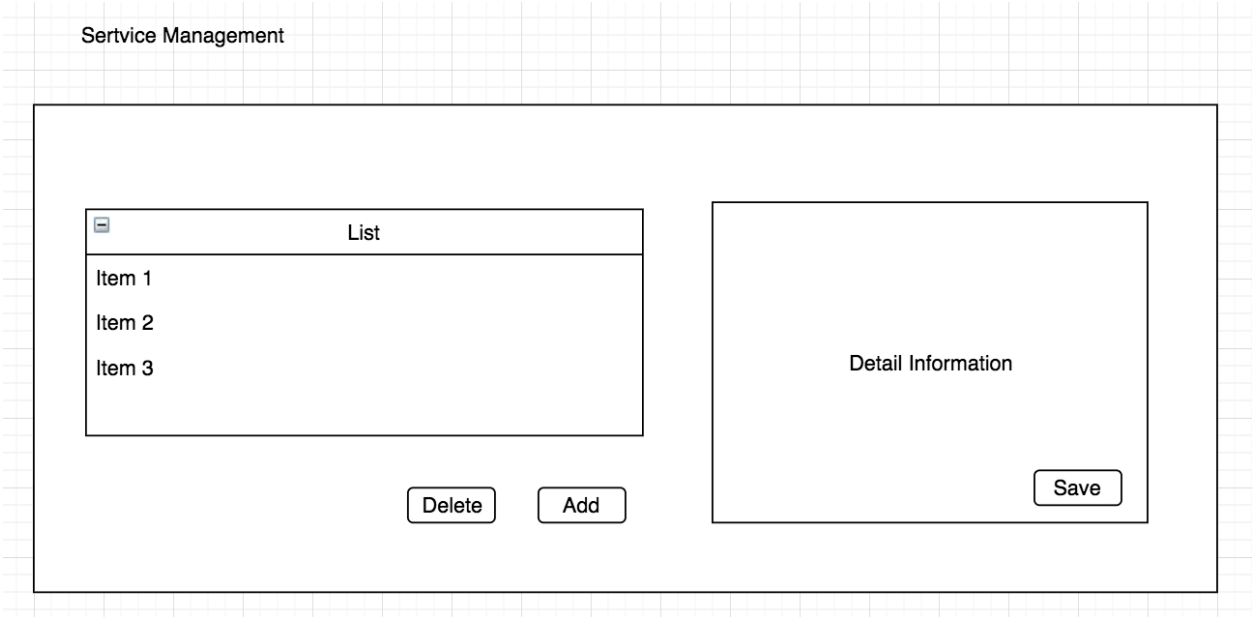
Sales Management



Staff Management



Service Management



Same as above, the left side shows all services, and the right-side panel shows the detail service information. User can add a new service or update the existing service. Click save will store the data to database.

Process Model

Software Development Process Models

Common Processing Activities

From 1970s to 2010s, the software development methodologies and processes changed a lot. Every organization has its own method to deal with the software development life cycle. Because of different idea of delivering software to production, they also have their working process that it ensures with how it gets the software development done.

Although there are so many methodologies and process in this world, we still can find the common processing activities, which are listed following:

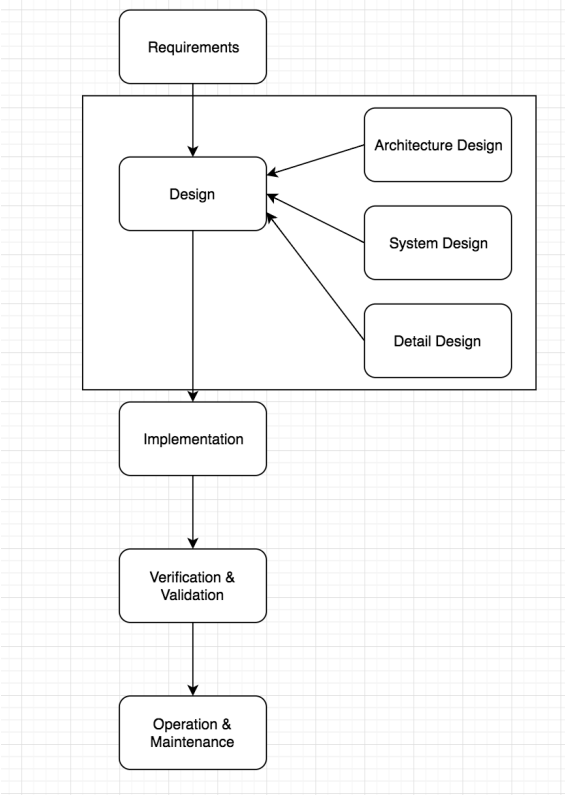
1. Market researching / Studying problems
2. Analysis of requirements
3. Design of system
    a. System specification
    b. User acceptance criteria
    c. Technology acceptance criteria
    d. Work flows
4. Implementation
    a. Detail functionalities
    b. Unit testing
    c. Integration Testing
5. Testing
    a. System testing
    b. End to end testing
    c. Regression testing
    d. Smoking testing
6. Deployment
7. Maintenance

Since all methodologies have similar activities, it becomes a matter of arranging those activities, of processing the details, and settling the level of details in every activity.

Here are the most popular software development methodologies since 1970s.

Waterfall Model

Waterfall model is a linear software development process, which means it is a sequential approach and every process has pre-requirement. It will not go to next step until the previous step is solved. It also cannot be reversed by itself.



The principle of waterfall is:

- The movement from phase to phase is always forward
- Requirements must be validated
- Each phase has milestones and has output documentations
- Evolving all plans, schedules, budgets and implementation at one time
- Required approval to move on
- Testing cannot be processed until implementation is done

Waterfall model was very popular during the years from 1980s to 1990s. It has several benefits, which are

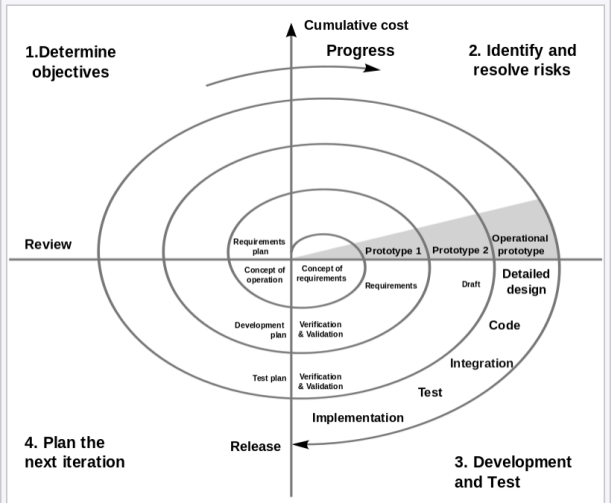
- Easy to understand because it is a linear structure
- The develop team knows the whole system very well
- Every phase is easily estimated and explicitly documented

Waterfall is often criticized as inflexible. The main reason is because of that irreversible forward motion. The requirements are also unknown sometimes until developers participate from the early phase. Based on the disadvantages of waterfall mode, most of organizations will do a kind of strategy to allow developers to revisit and revise decisions and documentations from their previous phases after moving on to next phase.

Since waterfall model won't revert the phases of software development, and the project 'Auto Dealer' is a short-term project, so I will not select waterfall as my software development methodology.

Spiral Model

Spiral model was published by Barry Boehm in 1988, which was combined with waterfall model and rapid prototyping methodologies. It is an iterative approach with a focus on risk management.



Spiral model (Boehm, 1988)

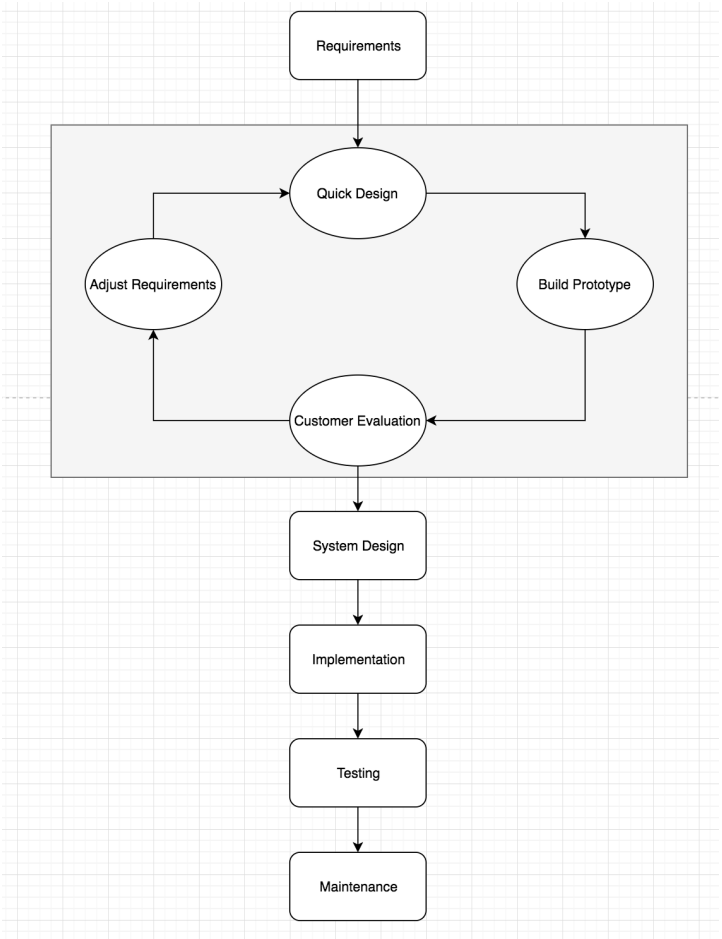
The spiral model has different phases with different activities. The first phase is to determine objectives, alternatives and constraints. For example, to define requirements and alternative thirty-party code; The second phase is to identify and resolve risks through these steps: evaluate identified alternatives, identify risks, resolve risks and product prototype. The next phase is to

develop and test. Same as waterfall, this phase will analyze performance of prototype and create or review design, implementation and testing. The last phase is to plan next iteration.

The spiral model is very flexible with good process visibility. The develop team focuses on the risk before all risks are identified. But since not every developer in team is expert of risk management, so this process model can be very slow and costly.

Back to our 'Auto Dealer' project, we neither have any existing builds, nor any member in this team has specialized skill of risk management, so this process model also is not option.

Prototyping



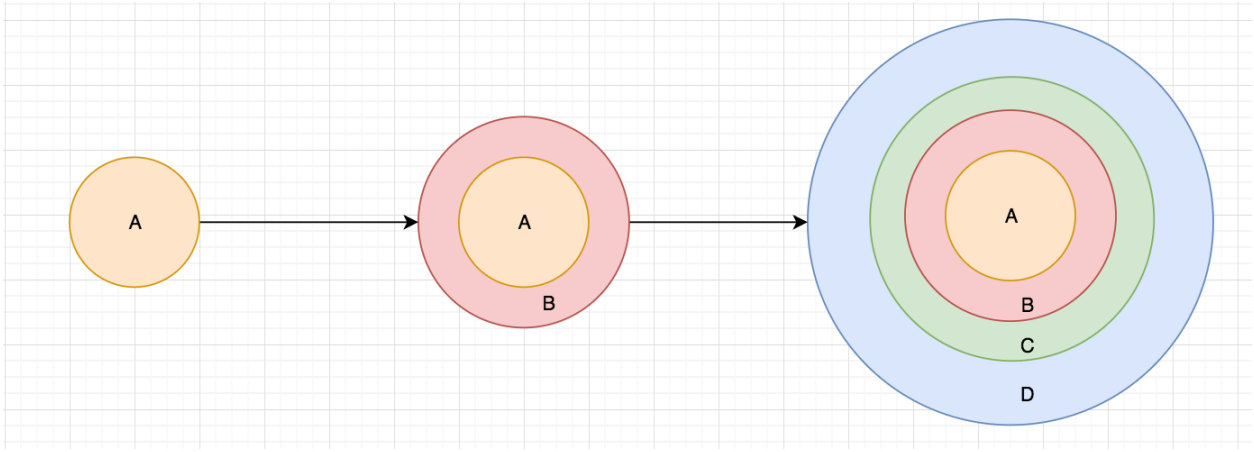
Before system design, prototyping model provides a rapid, short term cycle for customer to verify the requirements. Because some customers even don't know what their requirements are, this rapid prototyping model can confirm customer's idea and adjust the requirements if necessary. This is very useful and can reduce time and costs as well. The disadvantages of this process model are

also very obviously. First disadvantage is wasting time and money on implementing prototyping. Since some customers only have an idea from the beginning, the develop team needs several times of prototyping to get the final requirement. Second it is a little difficult for the develop team to think what level of the prototyping should be to fit the requirements. Developers may assume the final target based on their working experience.

This process model won't be a better choice for my current project. There's no customer which we need to demonstrate.

Incremental Model

The incremental model has a greater emphasis on producing intermediate versions, and each adding a small number of additional functionalities. The incremental model has advantages of flexibility, ability to explore poorly understood requirements and the implementation can be earlier. On the contrary, because of the poor process visibility, we are unable to schedule the plan very well.



First, the software development team only built and delivered to the customer with a few basic features. Then more and more iterations / versions are implemented and delivered to the customer until the desired system is realized.

Requirements of Software are first broken down into several modules that can be incrementally constructed and delivered. At any time, the plan is made just for the next increment and not for any kind of long-term plans. Therefore, it is easier to modify the version as per the need of the customer. Development Team first undertakes to develop core features (these do not need services from other features) of the system.

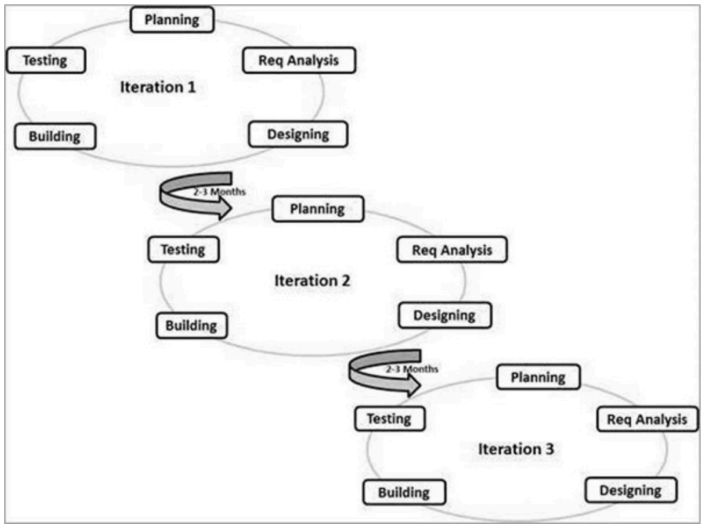
Once the core features are fully developed, then these are refined to increase levels of capabilities by adding new functions in Successive versions. Each incremental version is usually developed using an iterative waterfall model of development.

As each successive version of the software is constructed and delivered, now the feedback of the Customer is to be taken and these were then incorporated in the next version. Each version of the software has more additional features over the previous ones.

Agile Process Model

Agile process model is a modern variant iterative development which was introduced in 2001 with Agile Manifesto. It is a practice that promotes continuous iteration of development and testing throughout the software development lifecycle of the project. Both development and testing activities are concurrent. That means when develop team is implementing, the testing team also can do their testing work at the same time.

When working on a large project, the agile process can split the project into several individual models that designers work on. Since agile process is also based on iteration and incrementation mode, so the customer has early and frequent opportunities to look at the product and make decision or change the requirements. Here is the graphical illustration of the Agile Model:



As the chart above, Agile Method breaks the product into small incremental builds. Each iteration lasts from about one to three weeks. Each iteration involves cross functional team working simultaneously on various areas like planning, analysis, user accept criteria, time estimation,

design, coding, code review, unit testing and deployment. When the iteration is finished, the customer can get a working version from the develop team.

Since 2001, Agile Model has been the most popular software development model which is used by most of the organizations. Here are the reasons why.

Why chooses Agile

The benefits of using Agile are compelling. Before explaining the benefits of using Agile, let me explain the basic agile concept. First, in the early years, the organization hires people like product manager, program analyst, developers and QA. This situation has been changed as the software developers become software engineers. Second, different from the other process model with long meeting and discussion every day, the team only spends 10 to 15 minutes, stands in a circle and each team member talks about what they have done yesterday, what are they doing today and what they plan to do tomorrow. In this mode, the whole team could be aware of everything that assigned to each team member. It is much harder to make the team to take a wrong path. During the scrum every day, decisions can be made very quickly. Besides, the iteration of Agile model is very short. The iteration, which is called sprints, only takes one to three weeks. Each sprint contains:

- A sprint plan to decide what functionality will be built and deployed to the customer.
- The user stories which from backlog or created by the team member will apply to each sprint and assign to team.
- Before starting a new sprint, the software engineers need to gloom the tasks and give all tasks an estimation.
- During the implementation, software engineers need to make a demonstration of the working code to the team and product manager.
- A “retrospective” will be held to discuss what went well during the sprint and what need to be done better in the next sprint when that sprint is almost finished.

So, the benefits are also very obviously:

- **BUSINESS BENEFITS** Since the sprint in Agile is very short, so the customers and management team can get the production very quickly and frequently.
- **TEAM BENEFITS** During the daily scrumming meeting, sprint planning, retrospective meeting and sprint review, all team member includes software developers, testers and product managers are all getting benefits from it.

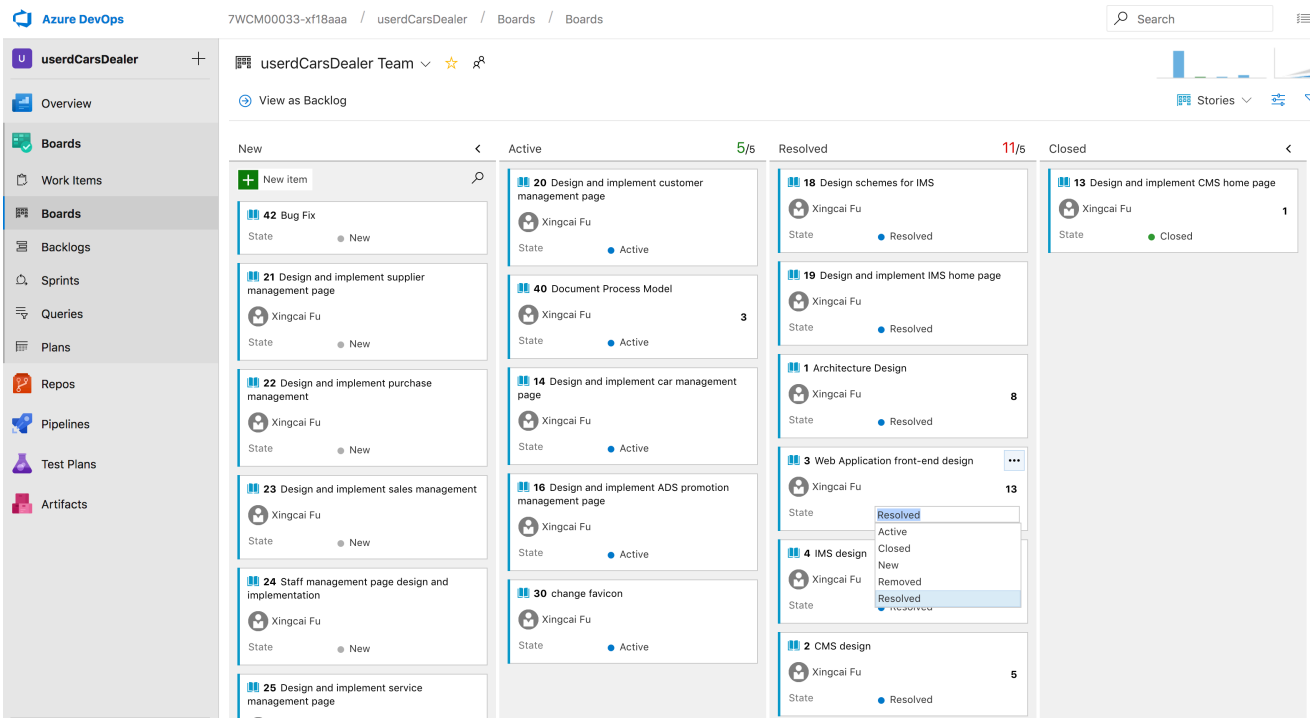
- **PRODUCTION BENEFITS** By breaking down the project into manageable units, the whole team can focus on high-quality development, testing, and collaboration. Since the production is built and deployed very frequently, quality is improved by finding and fixing defects quickly and identifying expectation mismatches earlier.

Apply Agile to Auto Dealer Application

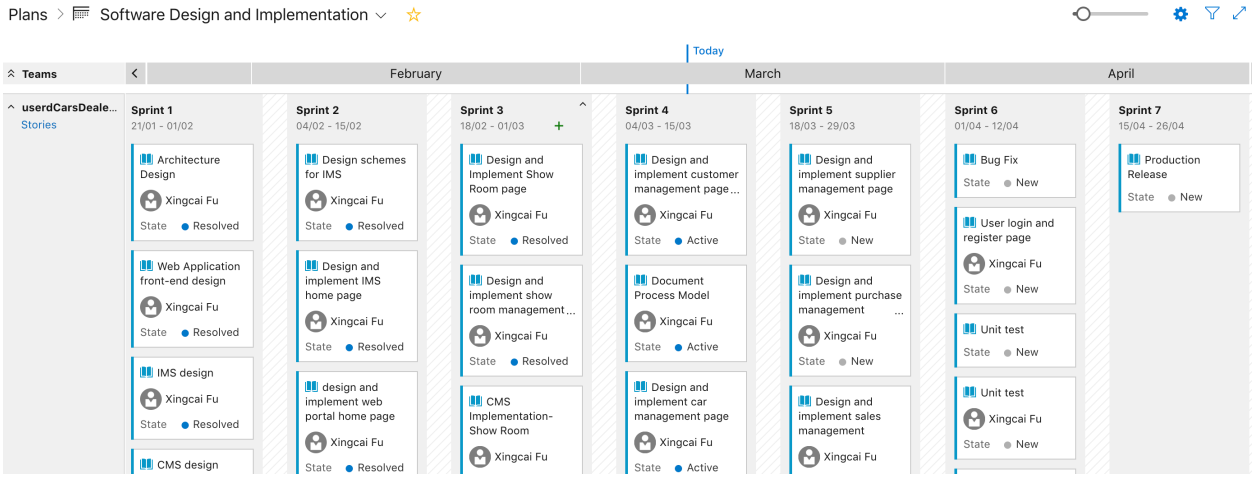
I have been using Agile for several years. I understand how many benefits that Agile brings to my team. As an experienced Agiler, I started to make sprint plan once I got this project. Since I am the only member in this team, I'm not only a product manager, tech lead, and also a tester. As a product manager, I split the whole development into seven sprints, with two weeks each. During the first two sprints, I did some work of high-level architecture design and scratch UI. In the middle of three sprints, I will do system design and implement all modules, such as web portal UI design, IMS/ CMS database design, and backend API implementation. I will do system testing and bug fix for the last two sprints. Though I applied Agile to the whole software development life cycle in each sprint, I still want to focus on different activities in different phases.

There are several applications as an Application Lifecycle Management (ALM) Suites can apply Agile model to software development team. The first one is JIRA Software, which is a tool developed by Atlassian and is use by the agile and development teams to plan, track the projects and their corresponding issues. Since JIRA basically is for business and not free, so I will continue with the next one. The second is Microsoft Team Foundation Server. It is an integrated suite of developer tool, build system, metrics, version control that are used by the specialized or qualified teams for organizing and running the projects. Azure is a DevOps tools which was Visual Studio Team Services. It provides multiple services which are Azure Pipelines, Azure Boards, Azure Artifacts, Azure Repos and Azure Test Plans. I select this tool because it's free for personal use.

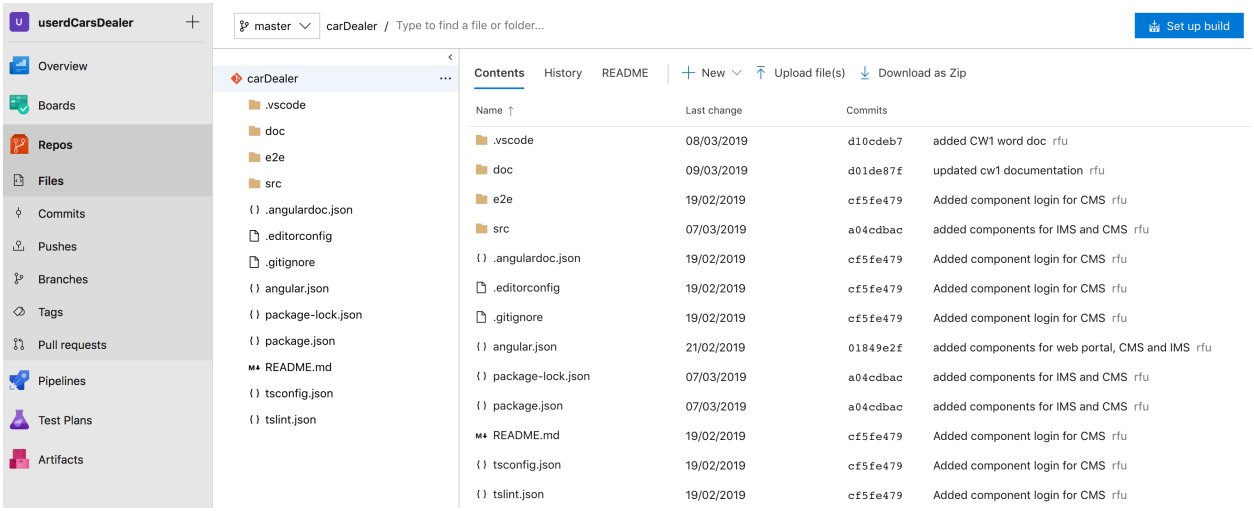
I use Azure to track my work with Kanban boards, backlogs, team dashboards and custom reporting. Here's an example of board for my current project.



And here's one part of my project plans for used car dealer.



It also integrated git repository and very easy for version controlling.



Any other better process model

On May 10th, 2018 one of the authors of the agile manifesto stated publicly that "Developers Should Abandon Agile". The author, Ron Jeffries, says that **Agility** is better than Agile. So, what is Agility and how Agility Boards worked? Comparing to Agile which has a bunch of built-in process, such as planning, review, design and retrospective, Agility is a much lighter model. Agility starts from a minimalist board which allows every team member to create their own workflow, meaning team members spend less time in project settings and focus more on the work.

Demian Entrekin, one of the Forbes Councils and CTO at Bluescape, had a report named 'Agile Is Not Enough'. In that report Demian has several interesting ideas. He admitted that Agile has won the battle of software development process model. But he thinks that though Agile makes our implementation faster, it doesn't make our products more innovative. He has two points, one is the Agile model makes the project not very clear and hard to see when the project can be finished. He thinks that faster sometimes doesn't mean better. The second point from Demain is management team and business leaders should embraced the concept of design thinking.

References

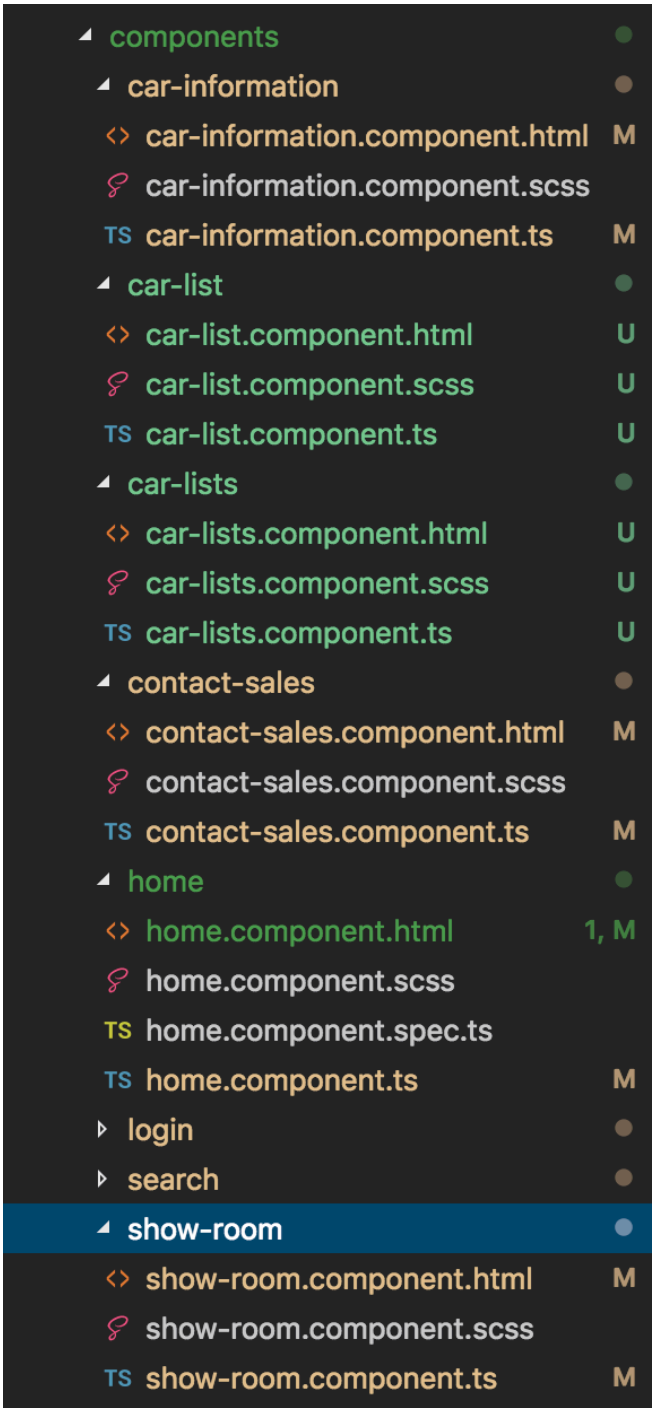
1. Wikipedia, the free encyclopedia, March 1st, 2019
   https://en.wikipedia.org/wiki/Software_development_process
2. Wikipedia, the free encyclopedia, March 1st, 2019
   https://en.wikipedia.org/wiki/Software_prototyping

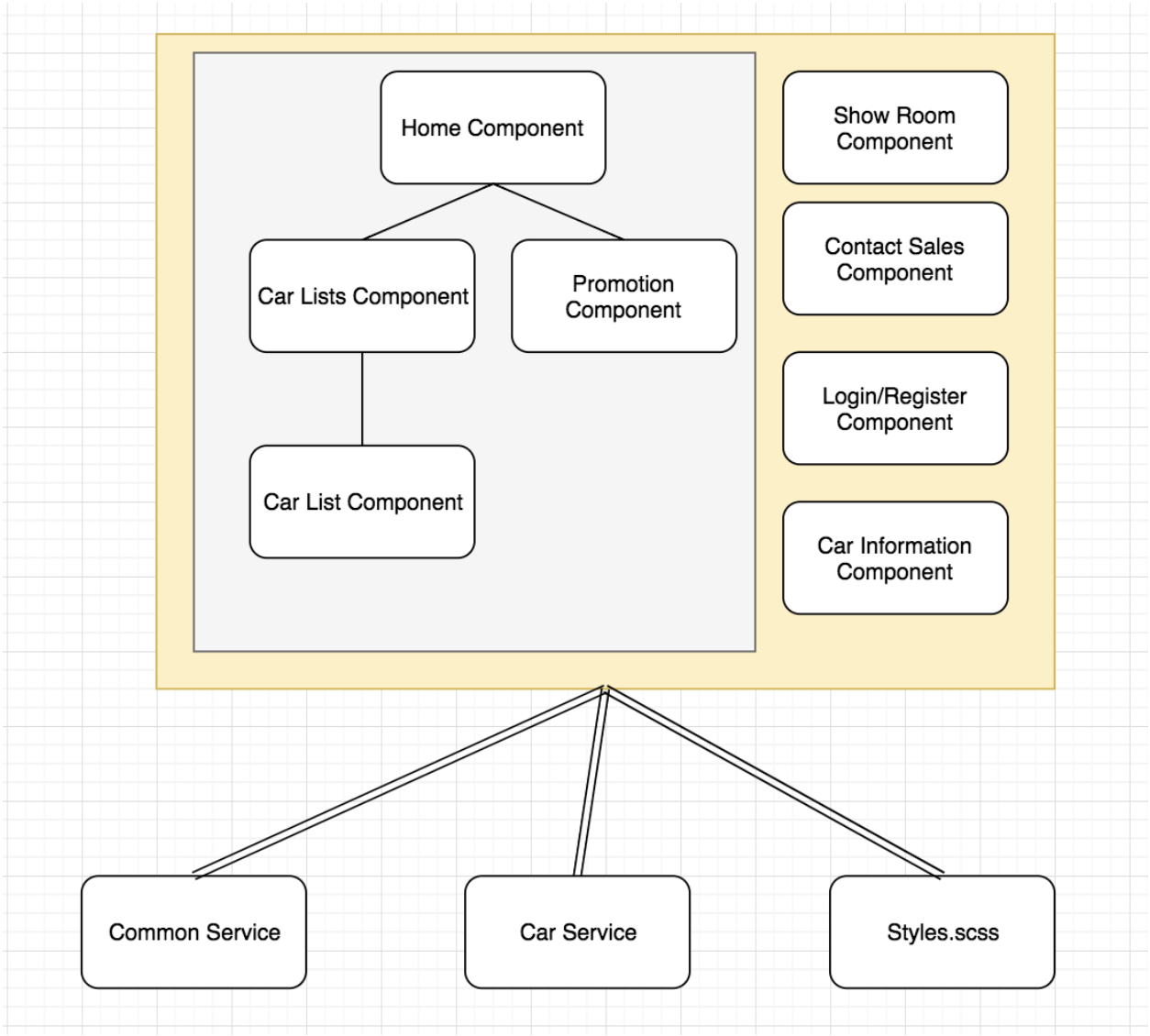
3. Tutorials point, SDLC – Agile Model
   https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm
4. Nicolas Quartieri, tech lead at Globant, October 28, 2014,
   https://stayrelevant.globant.com/en/5-reasons-to-choose-agile
5. MAX REHKOPF, Product Marketing Manager, Jira Software, August 29, 2018,
   https://www.atlassian.com/blog/jira-software/why-agility-is-better-than-agile
6. Demian Entrekin, Forbes Councils, CTO at Bluescape, May 19, 2017, 09:00AM,
   https://www.forbes.com/sites/forbestechcouncil/2017/05/19/agile-is-not-enough/#17a7804f4bfb
7.

Implementation

Web Portal

General Design



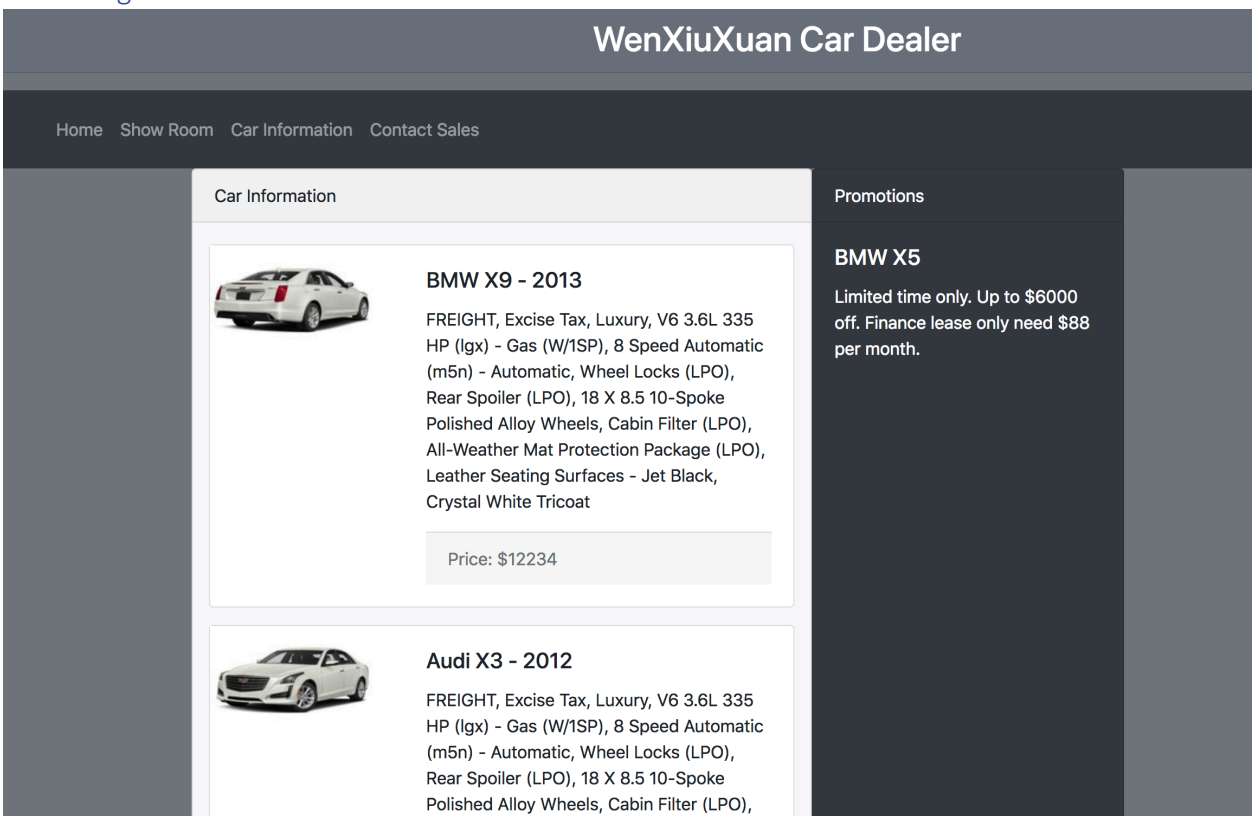


In the home page of Web Portal, it shows a list of car information, which is CarListsComponent. That component contains another sub-component CarListComponent, which shows one line of car information. The CarListsComponent and CarListComponent also can be used in another page where shows the car information.

The home page also includes other components for show room, contact sales, car information and user login component. There are several Angular services such as CommonService and CarService. All components in web portal pages are sharing the same CSS file styles.scss. There will be more common services in the future sprint.

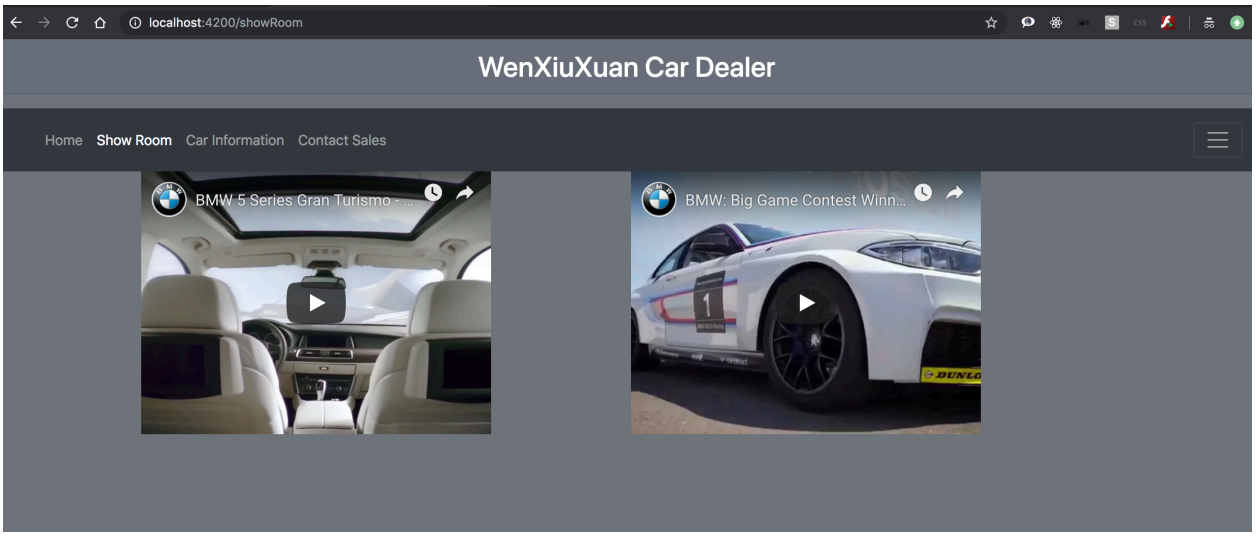
UI Design

Home Page

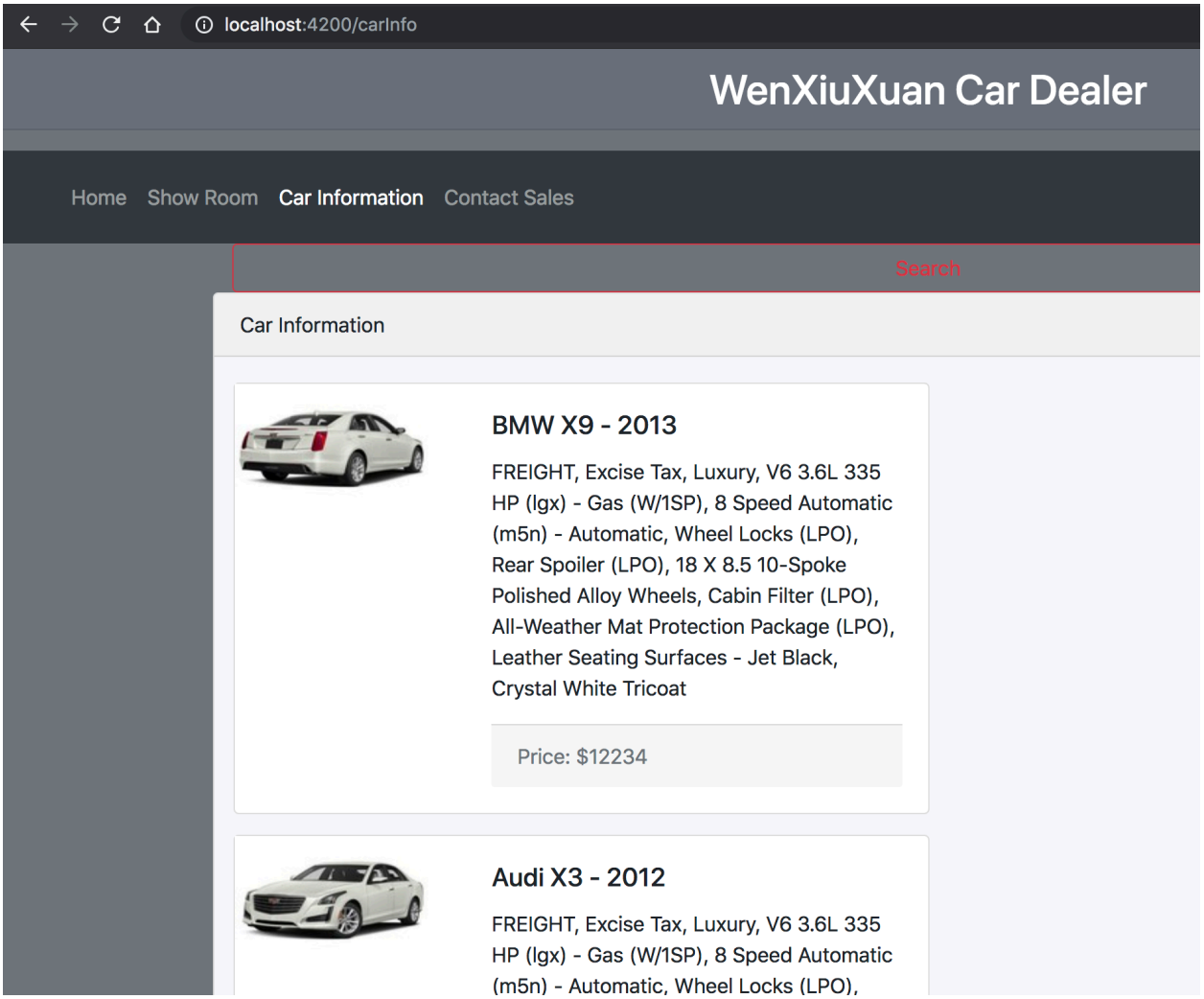


Just like the description above, this page shows car information as a list. The right side panel shows the current promotions or advertisements.

Show Room

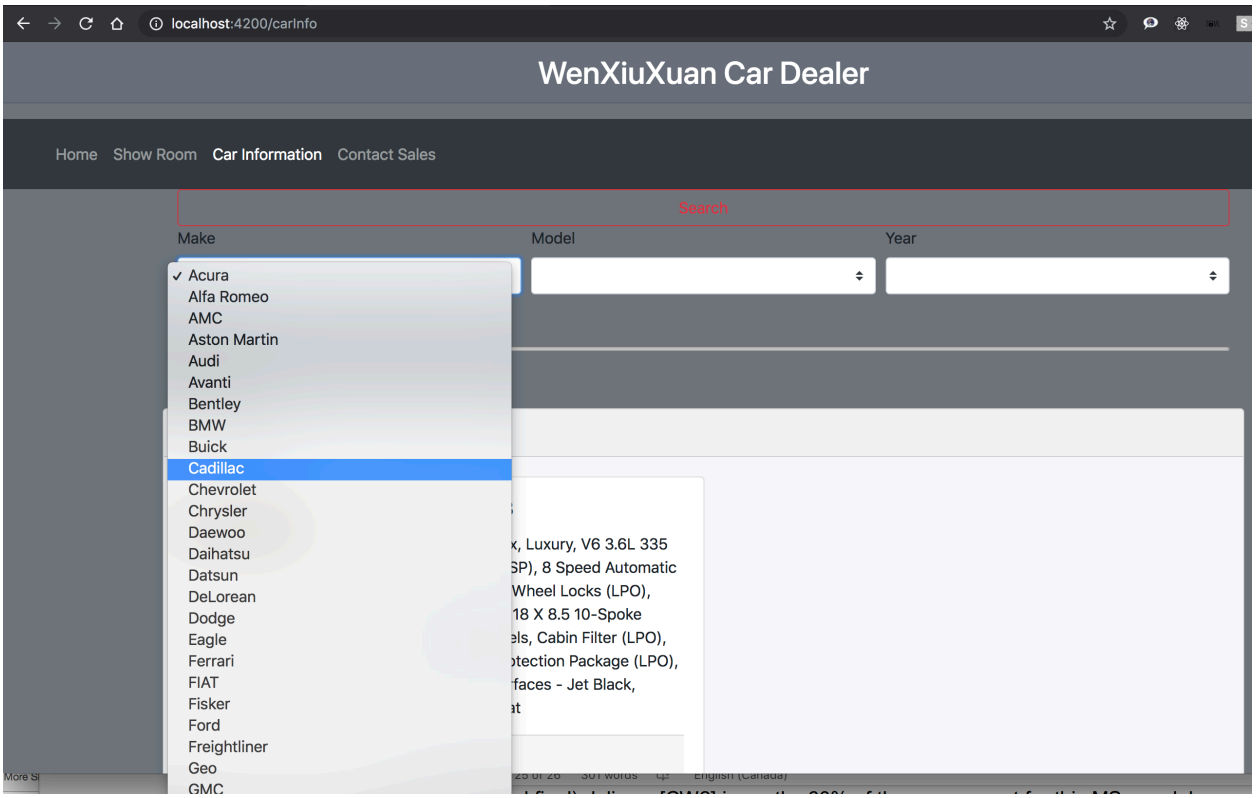


Car Information

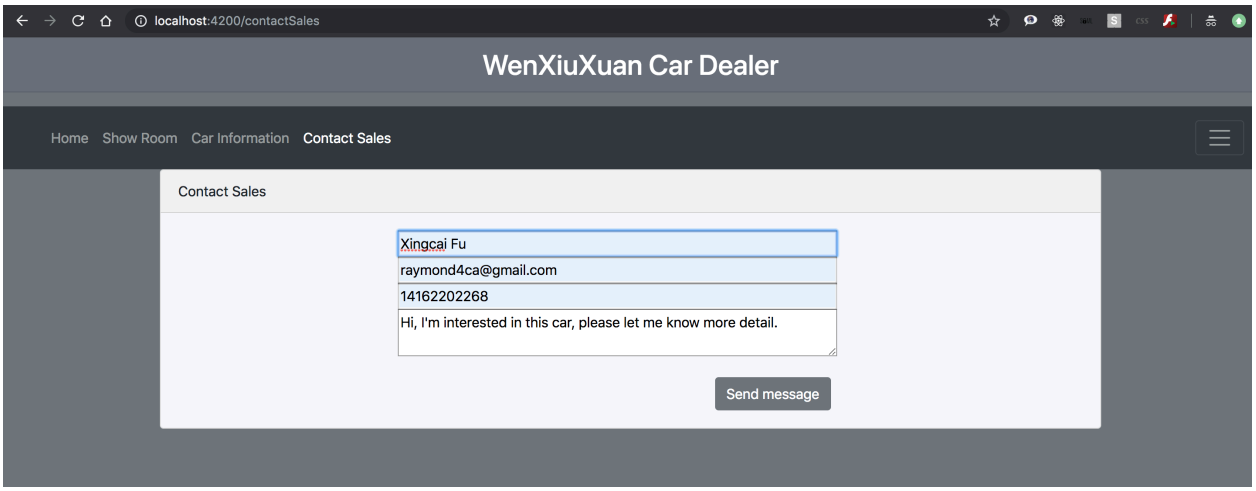


With Search Page

The search panel contains car searching conditions include car make, model, year, price and lots others. After searching, the results will be shown as lists in the left side.



Contact Sales



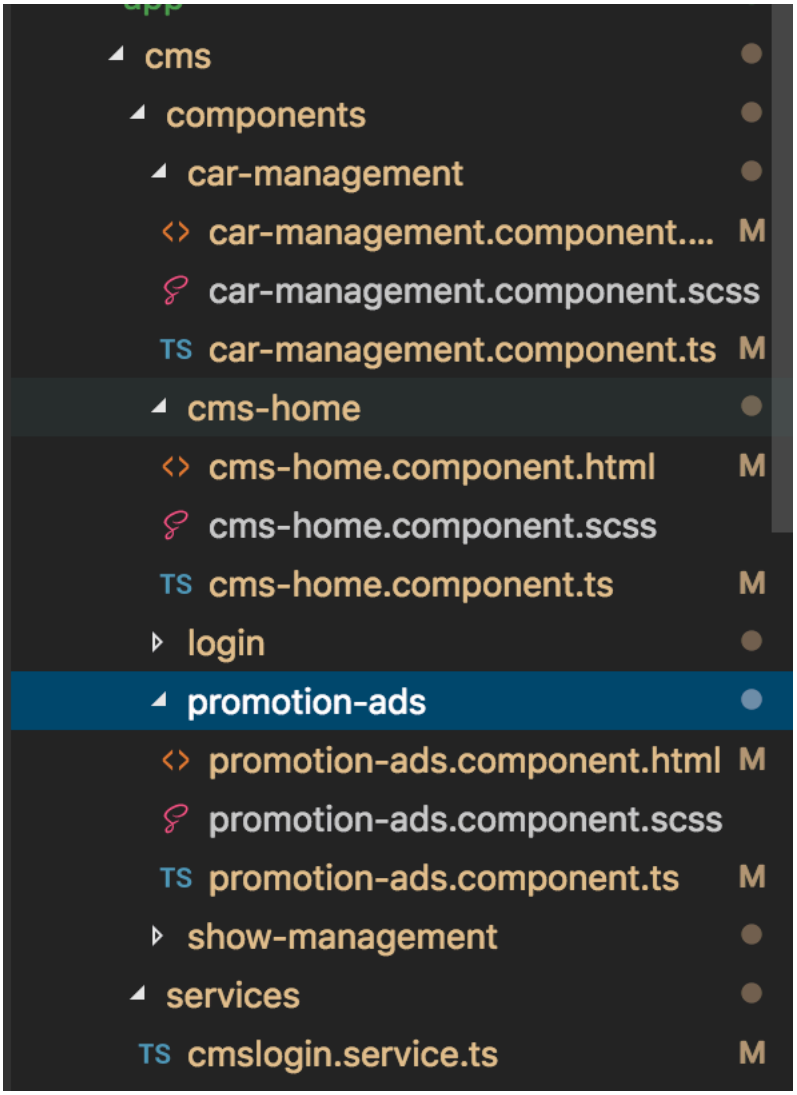
Common Service

This service provides common functionalities which contains loadImage, set/get Users, set/get Login Information, etc.,

Components

As the chart above, Web Portal includes HomeComponent, CarInformationComponent, ShowRoomComponent, ContactSalesComponent and LoginComponent. Home Component also contains CarListsComponent and PromotionComponent (will be done in next sprint).

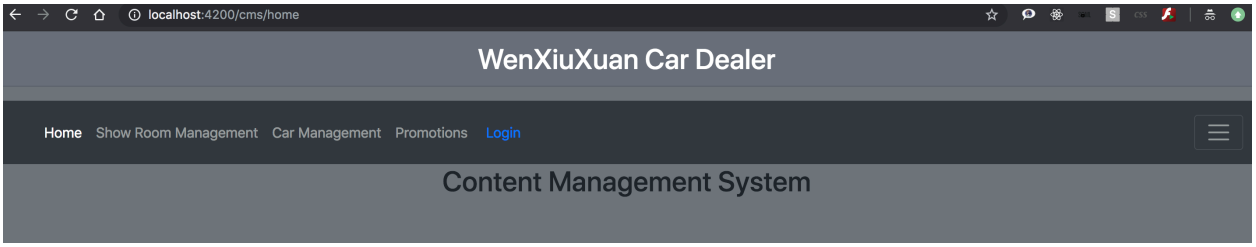
CMS



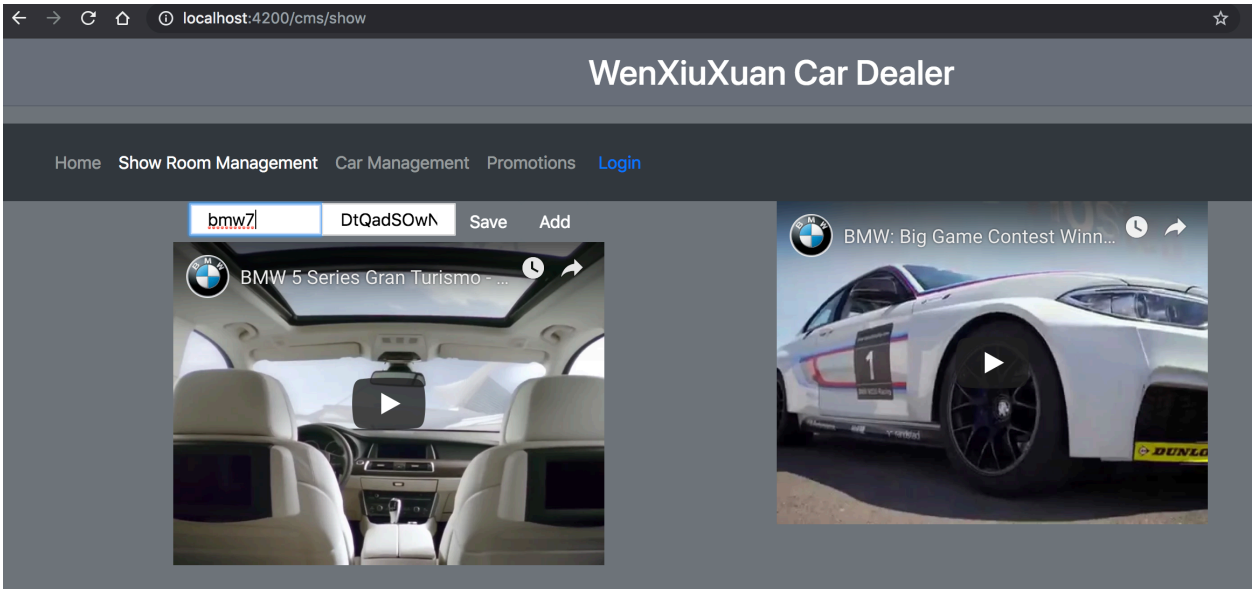
The CMS includes CarManagementComponent, ShowRoomManagementComponent, PromotionManagementComponent and LoginComponent. Also, several services like CmsLoginServie, CmsCommonService and more will be as part of CMS.

UI Design

Home Page

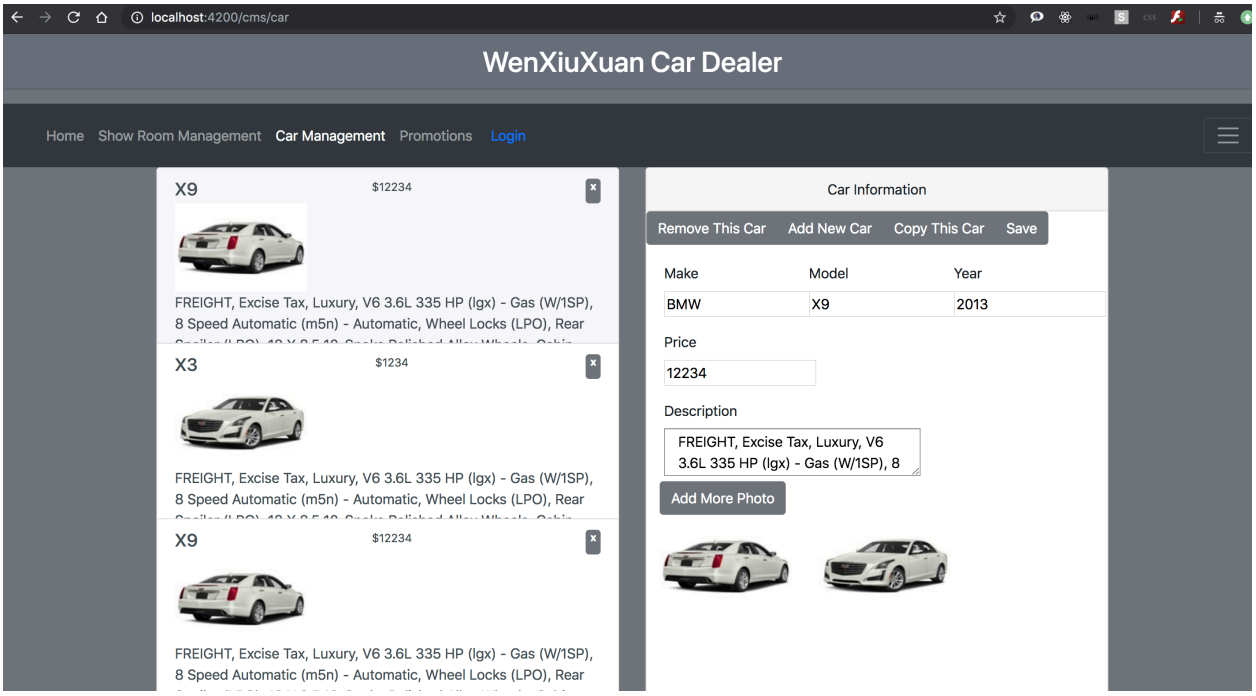


Show Room Management

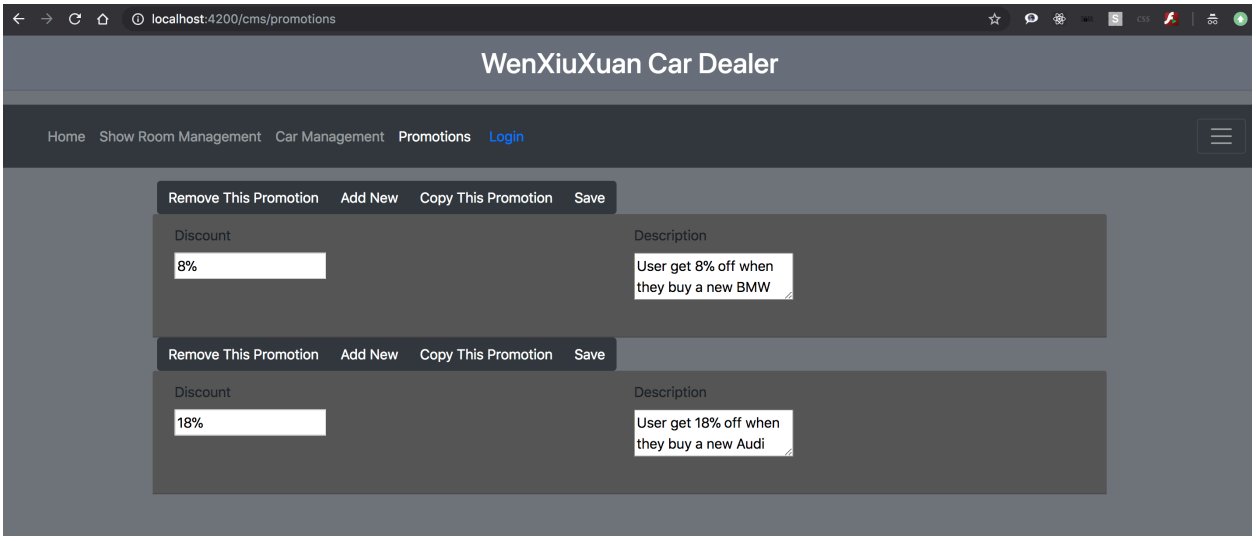


Car Management

When user selects any car in the list of left side, all car information will be carried out to right side form. User can add or update these data or upload extra photos.



Promotion Management



DB Schemes

According to the architecture design above, CMS could have DB schemes carInfo, promotions, showroom and user.

Car Info

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const collection = 'carInfo';

let CarInfo = new Schema(
  {
    brand: {
      type: String,
      required: true,
    },
    model: {
      type: String,
    },
    year: {
      type: Number,
    },
    description: {
      type: String,
    },
    kilometers: {
      type: Number,
    },
    status: {
      type: String,
      enum: ['used', 'new'],
    },
```

```
    bodyType: {
      type: String,
    },
    engine: {
      type: String,
    },
    transmission: {
      type: String,
    },
    driveTrain: {
      type: String,
    },
    stockNumber: {
      type: String,
    },
    colorInt: {
      type: String,
    },
    colorExt: {
      type: String,
    },
    passengers: {
      type: Number,
    },
    doors: {
      type: Number,
    },
    fuelType: {
      type: String,
    },
    price: {
      type: Number,
    },
    images: {
      type: Array,
    },
  },
  {
    collection: collection,
  },
);

module.exports = mongoose.model(collection, CarInfo);
```

Promotions

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const collection = 'promotions';

let Promotions = new Schema(
  {
    description: {
      type: String,
      required: true,
    },
    discount: {
      type: Number,
    },
    createdBy: {
      type: String,
    },
    createdOn: {
      type: Date,
      default: Date.now,
    },
    status: {
      type: String,
      enum: ['active', 'inactive'],
      default: 'active',
    },
  },
  {
    collection: collection,
  },
);

module.exports = mongoose.model(collection, Promotions);
```

Show Room

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const collection = 'showRoom';

let ShowRoom = new Schema(
  {
    name: {
      type: String,
      required: true,
    },
```

```
    videoUrl: {
      type: String,
    },
    startDate: {
      type: Date,
      default: Date.now,
    },
    endDate: {
      type: Date,
    },
  },
  {
    collection: collection,
  },
);

module.exports = mongoose.model(collection, ShowRoom);
```

User

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

let User = new Schema(
  {
    username: {
      type: String,
      required: true,
    },
    password: {
      type: String,
      required: true,
    },
    name: {
      type: String,
      default: this.username,
    },
    personalInfo: {
      type: Object,
    },
    createdBy: {
      type: String,
      default: 'admin',
    },
    createdOn: {
      type: Date,
      default: Date.now,
```

```
    },
    status: {
      type: String,
      enum: ['active', 'inactive'],
      default: 'active',
    },
    lastLogin: {
      type: Date,
      default: Date.now,
    },
  },
  {
    collection: 'user',
  },
);

module.exports = mongoose.model('User', User);
```

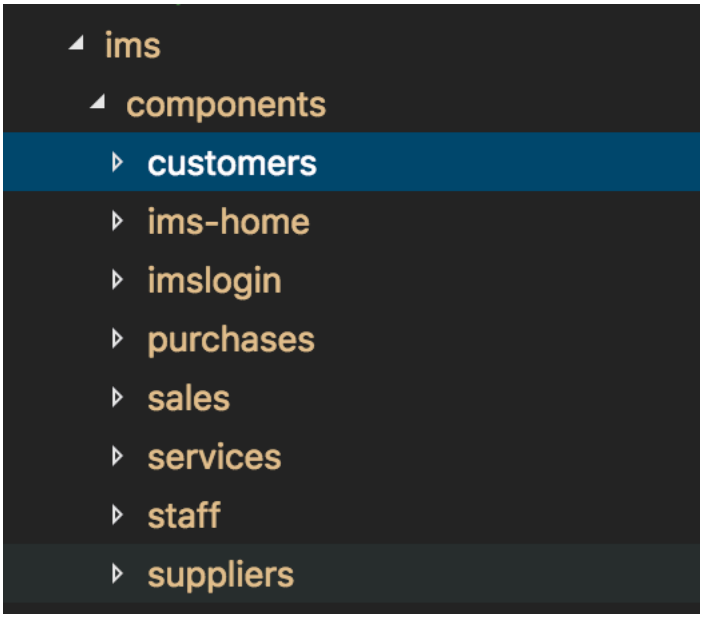
Routers

```
const carMakeRoutes = express.Router();
carMakeRoutes.get('/findAll', routeControl);

const carInfoRoutes = express.Router();
carInfoRoutes.post('/add', routeControl);
carInfoRoutes.get('/findAll', routeControl);
carInfoRoutes.post('/search', routeControl);
carInfoRoutes.post('/update', routeControl);
carInfoRoutes.post('/remove', routeControl);

const showRoomRoutes = express.Router();
showRoomRoutes.post('/add', routeControl);
showRoomRoutes.get('/findAll', routeControl);
showRoomRoutes.post('/update', routeControl);
showRoomRoutes.post('/remove', routeControl);

const eventRoutes = express.Router();
eventRoutes.post('/add', routeControl);
eventRoutes.get('/findAll', routeControl);
eventRoutes.post('/remove', routeControl);

const userRoutes = express.Router();
userRoutes.post('/register', routeControl);
userRoutes.post('/login', routeControl);
userRoutes.post('/logout', routeControl);
userRoutes.post('/add', routeControl);
```

```
userRoutes.get('/findAll', routeControl);
userRoutes.get('/findAllToday', routeControl);
userRoutes.get('/findHistoryToday', routeControl);
userRoutes.post('/remove', routeControl);
```

APIs

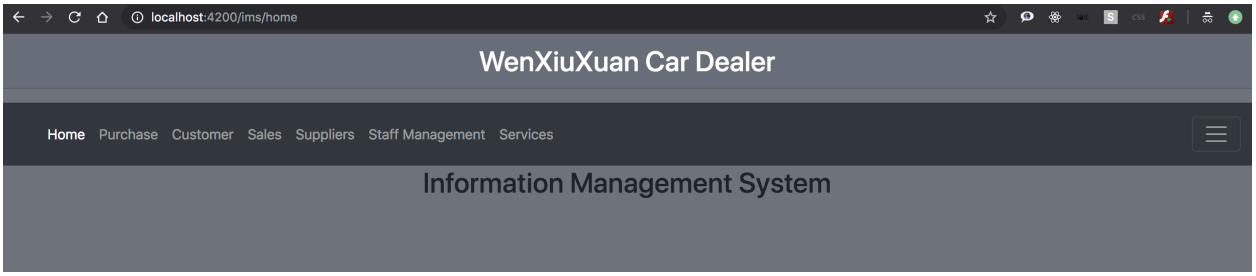
IMS



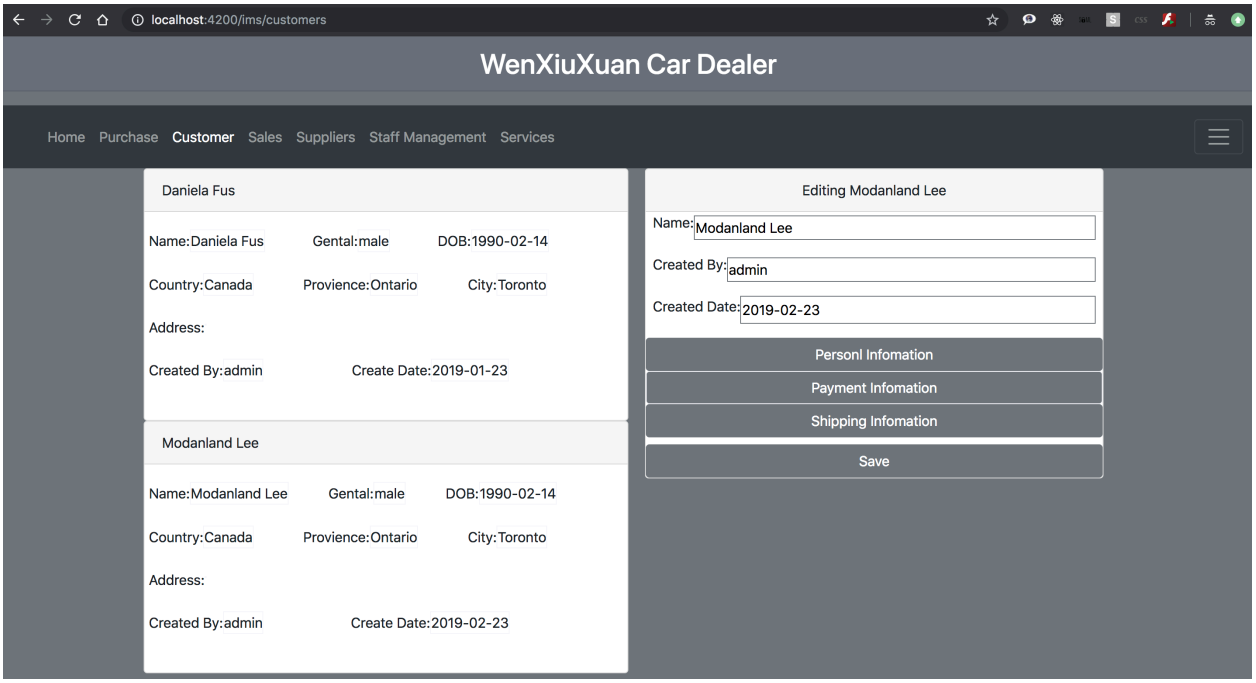
IMS includes multiple components like customers, home, login, purchases, sales, services, staff and suppliers. It also has several common services which will share in IMS module.

UI Design

Home Page

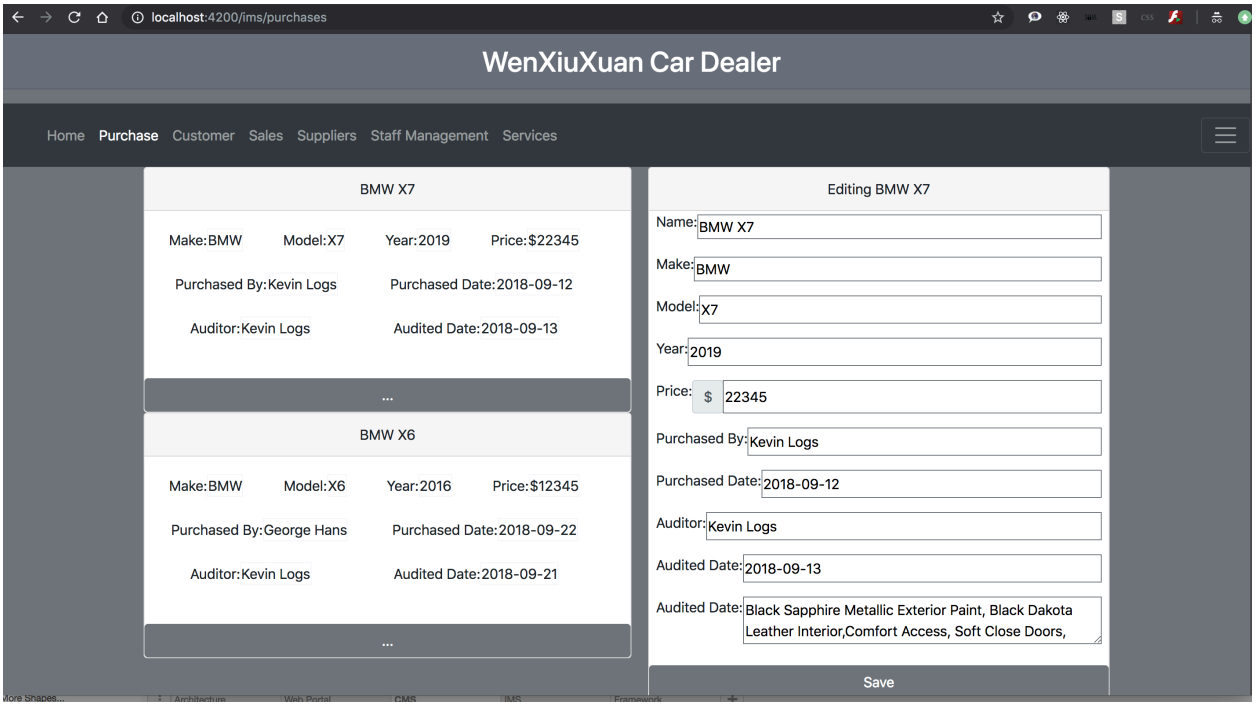


Customer Management

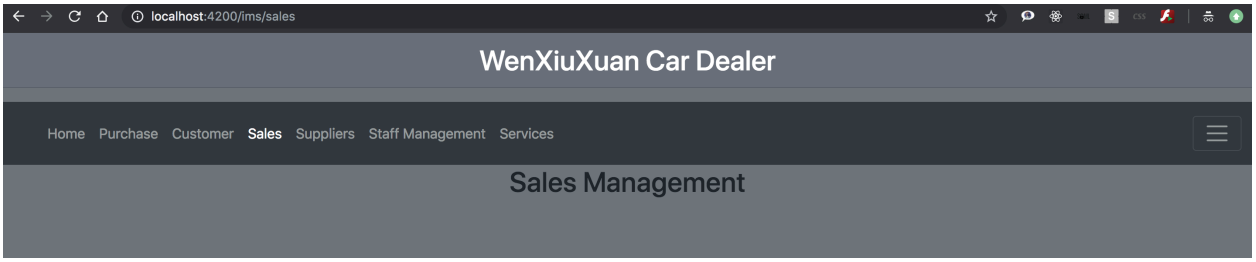


On the right side shows customer information and the staff can update it. The personal information includes first name, last name, gentle, DOB etc. The payment information panel includes credit card type, car number, security code, expire date and owner's address information. The shipping information panel contains the address for shipping, which could be different from payment.

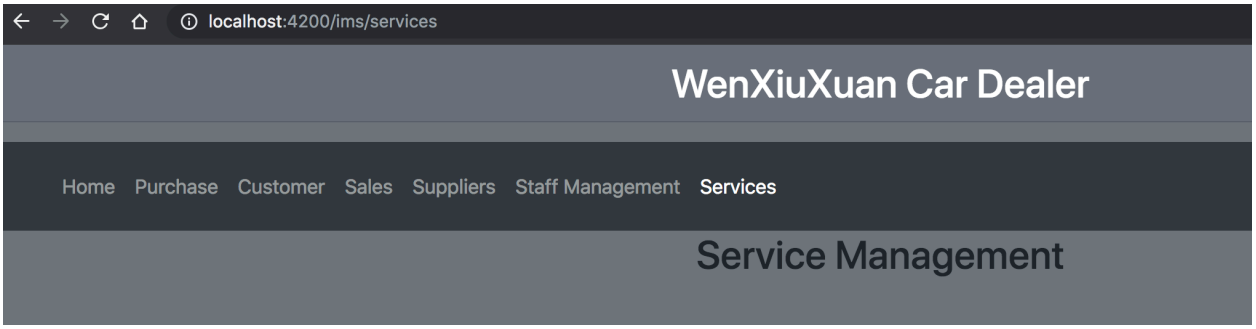
Purchase Management



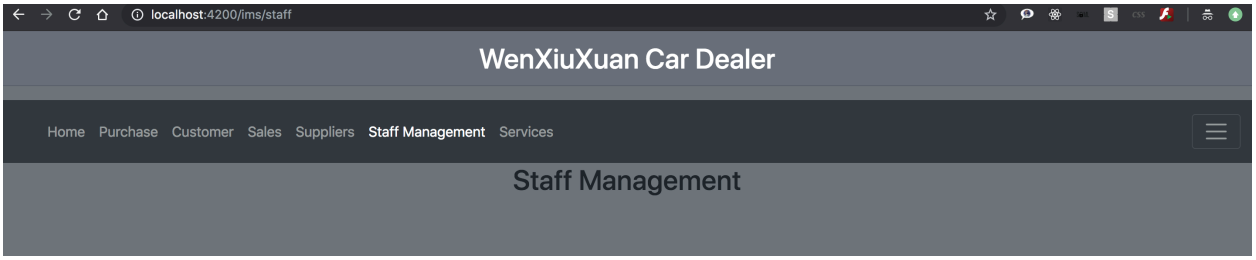
Sales Management (to be continued)



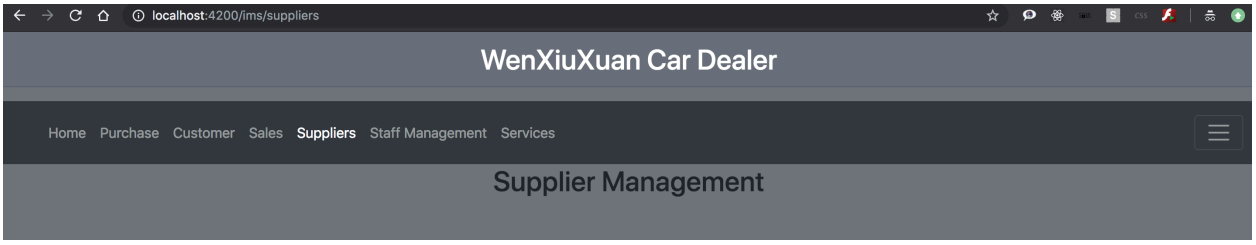
Service Management (to be continued)



Staff Management (to be continued)



Supplier Management (to be continued)



DB Schemes

Purchases

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const collection = 'purchases';

let Purchases = new Schema(
  {
    name: { type: String, required: true },
```

```
    description: { type: String },
    brand: { type: String },
    model: { type: String },
    year: { type: Number },
    price: { type: Number },
    purchasedOn: { type: Date, default: Date.now },
    purchasedBy: { type: String, default: 'Carlos Fuson' },
    createdBy: { type: String },
    createdOn: { type: Date, default: Date.now },
    auditedBy: { type: String },
    auditedOn: { type: Date, default: Date.now },
    supplierId: { type: String },
    status: { type: String, enum: ['active', 'inactive'], default: 'active' },
  },
  {
    collection: collection,
  },
);

module.exports = mongoose.model(collection, Purchases);
```

Customers(Users)

```
let User = new Schema(
  {
    username: { type: String, required: true },
    password: { type: String, required: true },
    name: { type: String, default: this.username },
    personalInfo: { type: Object },
    createdBy: { type: String, default: 'admin' },
    createdOn: { type: Date, default: Date.now },
    status: { type: String, enum: ['active', 'inactive'], default: 'active' },
    lastLogin: { type: Date, default: Date.now },
  },
  {
    collection: 'user',
  },
```

Sales

```
let Sales = new Schema(
  {
    username: { type: String, required: true },
    password: { type: String, required: true },
    name: { type: String, default: this.username },
    createdBy: { type: String, default: 'admin' },
    createdOn: { type: Date, default: Date.now },
    status: { type: String, enum: ['active', 'inactive'], default: 'active' },
```

```
    lastLogin: { type: Date, default: Date.now },
  },
  {
    collection: collection,
  },
```

Suppliers

```
let Suppliers = new Schema(
  {
    name: { type: String, required: true },
    description: { type: String },
    address: { type: String },
    phone: { type: String },
    email: { type: String },
    createdBy: { type: String, default: 'admin' },
    createdOn: { type: Date, default: Date.now },
    status: { type: String, enum: ['active', 'inactive'], default: 'active' },
  },
  {
    collection: collection,
  },
```

Staffs

```
let Staffs = new Schema(
  {
    username: { type: String, required: true },
    password: { type: String, required: true },
    name: { type: String, default: this.username },
    createdBy: { type: String, default: 'admin' },
    createdOn: { type: Date, default: Date.now },
    status: { type: String, enum: ['active', 'inactive'], default: 'active' },
    lastLogin: { type: Date, default: Date.now },
  },
```

Services

There will have several services to deal with all scheme data.

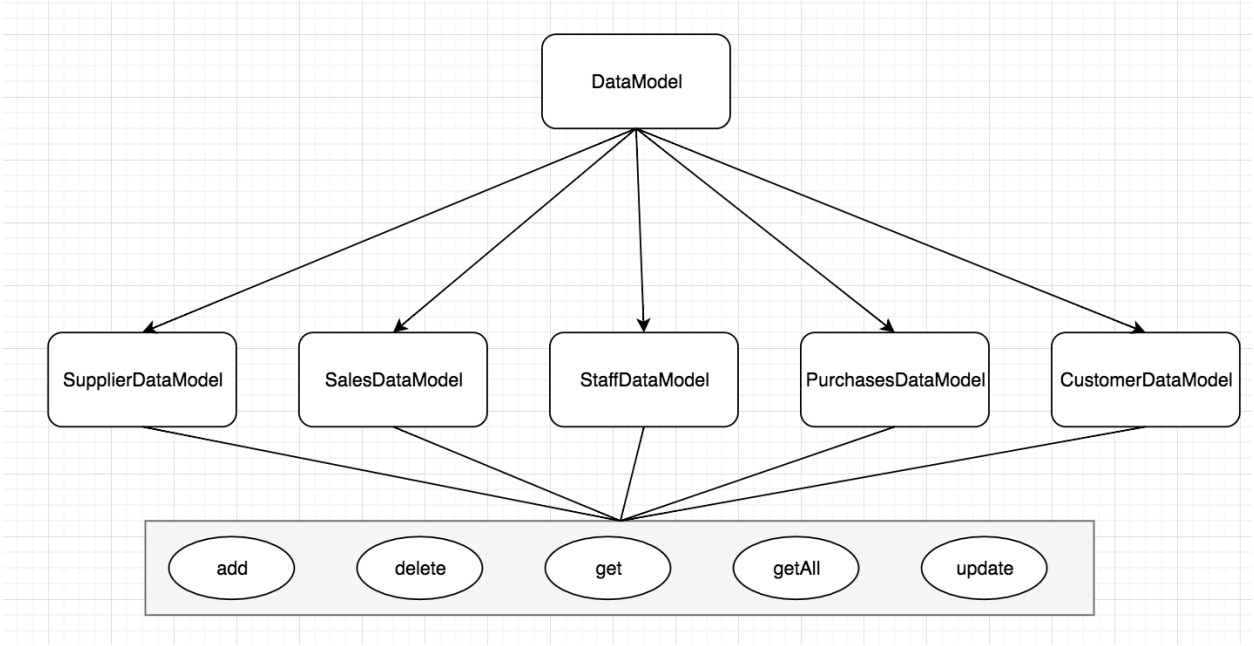
Routers

Routers to processing HTTP request for IMS are:

- Purchases:
  - POST: add/delete
  - PATCH: update
  - GET: getAllPurchases
- Suppliers:
  - POST: add/delete
  - PATCH: update
  - GET: getAll
- Sales/Users/Staff:

POST: add/delete/setPermission
PATCH: update
GET: getAll/getPermission/getSales

APIs



I'm using a base data model class. All HTTP request actions through different routers will find data model class automatically by the following code.

```
function routeCtl(req, res) {
  console.log('control: ', req.baseUrl, req.path, req.body);
  let name = req.baseUrl.substr(1);
  console.log('name:', name);
  let obj;
  switch (name) {
    case 'ims/purchases: {
      obj = new ImsPurchasesDataModel(req, res);
      break;
    }
    case 'ims/sales: {
      obj = new ImsSalesDataModel(req, res);
      break;
    }
    case 'ims/staffs: {
      obj = new ImsStaffsDataModel(req, res);
      break;
    }
    case 'ims/suppliers': {
      obj = new ImsSuppliersDataModel(req, res);
      break;
```

```
    }
    case 'ims/customers': {
      obj = new ImsCustomersDataModel(req, res);
      break;
    }
    default: break;
  }
```