

# **Pawsup**

## **Systems Design Document**

### **CSCC01**

Instructor: Ilir Dema

Group Members: Gary Xie, Jonathan Bai, SuTong Kong, Ivan Shao, Meng Zhao, Jesse Zhang, Tyler Reichert

## **Table of Contents**

CRC Cards.....	2
System Interaction Description.....	9
System Architecture.....	9
System Decomposition.....	9

## CRC Cards

### Frontend

<b>Class:</b> App.js
<b>Responsibilities:</b> Controls the entire frontend and routing
<b>Collaborators:</b> HomePage.js SignInPage.js ServicePage.js ProductPage.js MediaPage.js

<b>Class:</b> SigninPage.js
<b>Responsibilities:</b> Sign-in page for users
<b>Collaborators:</b> HeaderMenu.js

<b>Class:</b> SignupPage.js
<b>Responsibilities:</b> Sign-up page for users
<b>Collaborators:</b> HeaderMenu.js

<b>Class:</b> FeaturedProducts.js
<b>Responsibilities:</b> Overview of the products offered
<b>Collaborators:</b> None

<b>Class:</b> FeaturedServices.js
<b>Responsibilities:</b> Overview of the services offered
<b>Collaborators:</b> None

<b>Class:</b> HomePage.js
<b>Responsibilities:</b> HomePage for users
<b>Collaborators:</b> HeaderMenu.js Footer.js HomePageSearch.js FeaturedServices.js FeaturedProducts.js

<b>Class:</b> HomePageSearchService.js
<b>Responsibilities:</b> Search Component for services based on Location, time, and pets.
<b>Collaborators:</b> None

<b>Class:</b> AccountPage.js
<b>Responsibilities:</b> Account Page for both the customers and the service providers.
<b>Collaborators:</b> HeaderMenu.js Footer.js

<b>Class:</b> ProductPage.js
<b>Responsibilities:</b> Product page for displaying goods to sell(food, toys)

**Collaborators:**

HeaderMenu.js

Footer.js

**Class: ServicePage.js**

**Responsibilities:**

Service page for displaying services to provide in various locations

**Collaborators:**

HeaderMenu.js

Footer.js

**Class: CommentSection.js**

**Responsibilities:**

Display List of Comments given the information from the backend.

**Collaborators:**

HeaderMenu.js

Footer.js

**Class: ServiceDetailPage.js**

**Responsibilities:**

Displays the list of details of each service.

**Collaborators:**

HeaderMenu.js

CommentSection.js

Footer.js

**Class: MediaDetailPage.js**

**Responsibilities:**

Displays the list of details of each media.

**Collaborators:**

HeaderMenu.js  
CommentSection.js  
Footer.js

**Class:** ProductDetailPage.js

**Responsibilities:**  
Displays the list of details of each product.

**Collaborators:**  
HeaderMenu.js  
CommentSection.js  
Footer.js

## Backend

**Class Name:** db

**Parent Class:** None

**Subclasses:** None

**Responsibilities:**

- Data access object which allows the user to connect to and query the PostgreSQL database.

**Collaborators:**

- None

**Class Name:** Model

**Parent Class:** None

**Subclasses:** UserModel, ServiceModel

**Responsibilities:**

- Abstract class with responsibility to manage the logic and data of the application. Contains basic methods applicable for all model extensions.

**Collaborators:**

- db

**Class Name:** UserModel

<b>Parent Class:</b> Model <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>Handles backend database logic related to Users.</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>db</li> </ul>

<b>Class Name:</b> ServiceModel	
<b>Parent Class:</b> Model <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>Handles backend database logic related to Services.</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>db</li> </ul>

<b>Class Name:</b> ProductModel	
<b>Parent Class:</b> Model <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>Handles backend database logic related to Products.</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>db</li> </ul>

<b>Class Name:</b> CommentModel	
<b>Parent Class:</b> Model <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>Handles backend database logic related to Comments.</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>db</li> </ul>

<b>Class Name:</b> MediaPageModel	
<b>Parent Class:</b> Model <b>Subclasses:</b> None	
<b>Responsibilities:</b>	<b>Collaborators:</b>

<ul style="list-style-type: none"> <li>• Handles backend database logic related to Media.</li> </ul>	<ul style="list-style-type: none"> <li>• db</li> </ul>
--	--

<b>Class Name:</b> AuthController	
<b>Parent Class:</b> None <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>• Handles the logic for HTTP requests to the /api/auth route.</li> <li>• User login authentication.</li> <li>• User session.</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>• None</li> </ul>

<b>Class Name:</b> ServiceController	
<b>Parent Class:</b> Model <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>• Handles the logic for HTTP requests to the /api/services route.</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>• None</li> </ul>

<b>Class Name:</b> ProductsController	
<b>Parent Class:</b> Model <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>• Handles the logic for HTTP requests to the /api/products route.</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>• None</li> </ul>

<b>Class Name:</b> UserController	
<b>Parent Class:</b> Model <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>• Handles the logic for HTTP requests</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>• None</li> </ul>



to the /api/user route.	
-------------------------	--

<b>Class Name:</b> CommentsController	
<b>Parent Class:</b> Model <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>Handles the logic for HTTP requests to the /api/comments route.</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>None</li> </ul>

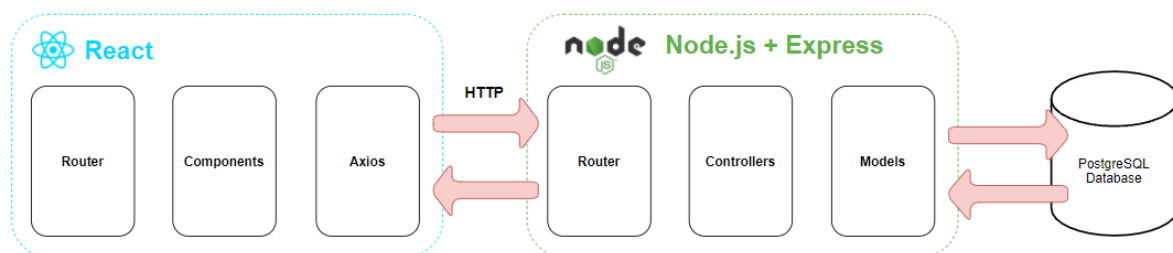
<b>Class Name:</b> Router	
<b>Parent Class:</b> None <b>Subclasses:</b> None	
<b>Responsibilities:</b> <ul style="list-style-type: none"> <li>Handles the routes for our Express server and connects them to the correct controller.</li> </ul>	<b>Collaborators:</b> <ul style="list-style-type: none"> <li>AuthController</li> <li>ServiceController</li> <li>UserController</li> </ul>

## System Interaction Description

Our project must be run on one of the following operating systems: Windows 10, MacOS, or Linux. **Node.js** (Version 14.7+) as well as **npm** must be installed in order to compile the front-end code and run the application server. For the database, **PostgreSQL** (Version 13+) must be installed and running. To allow users to connect to the application, the server must be hosted on a machine with adequate storage and bandwidth, alongside a domain name (website URL) registered in a DNS for users to access via a browser.

## System Architecture

We are using React and Bootstrap for our front-end, and it connects to the back-end server using Axios to send HTTP requests. In the backend, the router handles Requests and routes them to the correct Controller endpoint, which will use Models to communicate with the database (PostgreSQL) using a connection pool. Here is a diagram of our system architecture:



## System Decomposition

The user will access the React frontend client for our application by connecting to the correct URL via the Internet. The client will connect with our Node.js + Express server by using Axios to send HTTP requests to our REST API, such as for creating an account, logging in/out, searching for services or products, or making a purchase. When the backend server receives a request, it will process it and query the PostgreSQL database to update the stored data and send back relevant information. We are using a three-tier architecture to prevent users from directly accessing the database, for security purposes.

To handle errors and edge cases (such as empty user input for a required field), the frontend will prevent the user from sending an invalid request and show a descriptive error message to the user explaining what caused the error. In case a request with invalid data is sent to the server, the backend controller will validate that the data is correct before processing it to the database, and will reject the request if it is invalid.