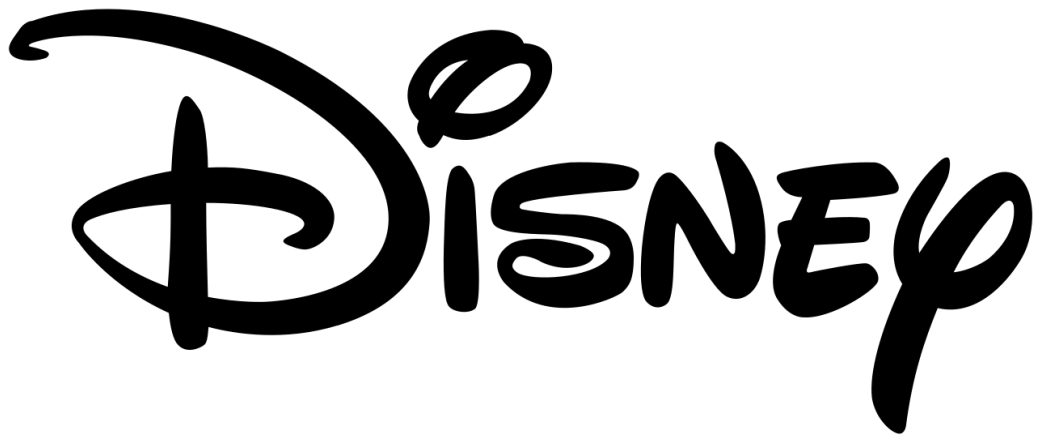
프로젝트기반 프론트엔드 웹 & 앱 SW 개발자 양성과정



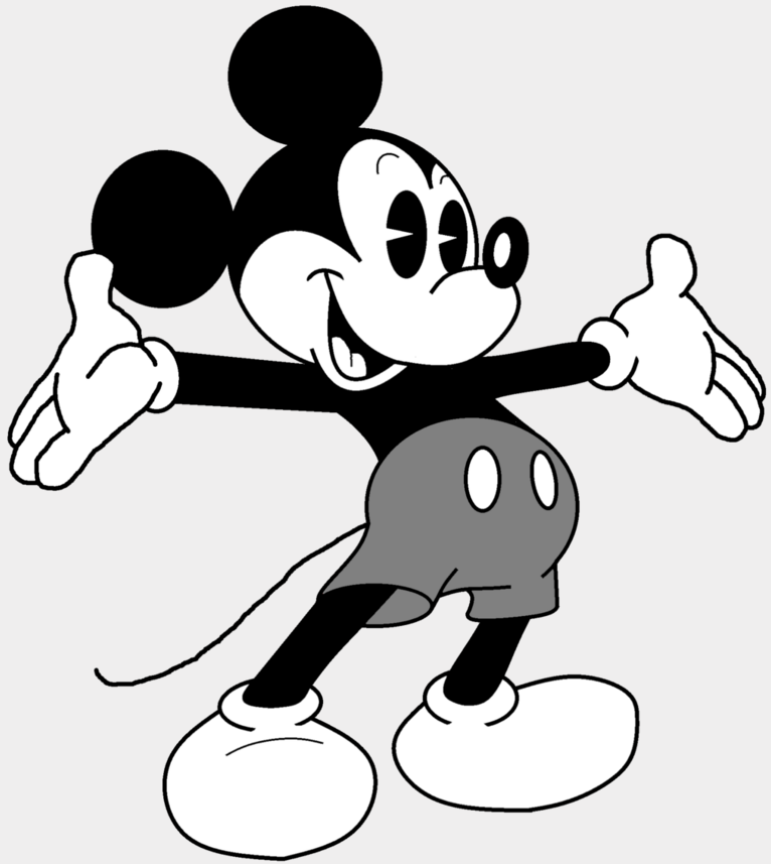
고소현

1 기획 의도

2 디자인 UI UX

3 주요 동작 및 기능

4 프로젝트 바로가기



# 자기 소개

INFJ

고소현

디즈니
좋아

쥐띠

# 프로젝트 기간

2024 JANUARY & FEBRUARY

| SUN | MON | TUE | WED | THU | FRI | SAT |
|---|---|---|---|---|---|---|
| | 1 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 | 2 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | | |

- 기획
- HTML & CSS
- VUE & JS
- PPT & 발표준비

# 개발 환경

Firebase

VS code와 Vue.js, Vue Router 그리고 Firebase를 사용해 웹앱 개발

# 사용 언어




HTML, CSS, JS를 사용

# 기획 의도


EUROPEAN HEADQUARTERS

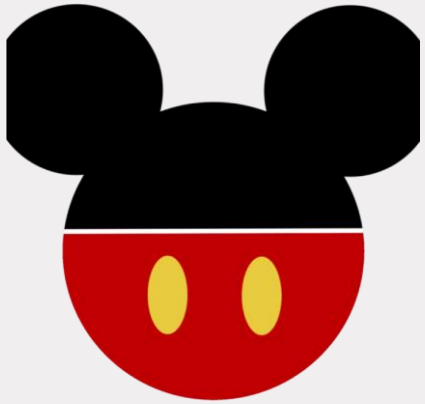
Directors :
GEORGE KAMEN, (U S A.)
Managing Director.
WILLIAM B. LEVY, (U.S.A.)
THOMAS SWAN, M.A., LL.B.

ASTORIA HOUSE.
62
SHAFTESBURY AVENUE,
LONDON, - W. I.
Telegraphic Address :
"MICKMOUSE."

디즈니를 정말 좋아하는 사람으로써 디즈니랜드, 디즈니리조트, 디즈니스토어를 통합해 한 번에 모든 것을 할 수 있으면 좋겠다고 생각하여 통합 웹앱을 만들어보았습니다.

# 디자인
# UI & UX

# 톤앤매너



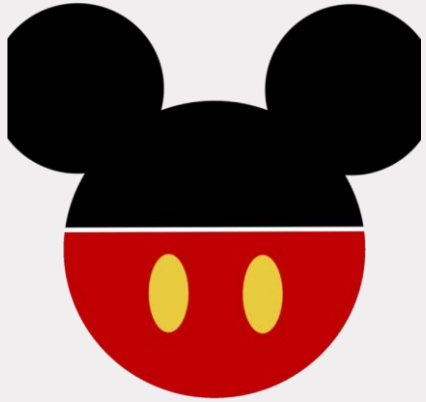
주조색

보조색

보조색

보조색

# 톤앤매너



웹 폰트

Font-family : 'Pretendard-Regular'

Font-weight : 400

Font-style : normal

폰트 사용 이유 :

가장 보편적이고 사용자가 보기 편한 폰트를 사용

DISNEY MICKEYMOUSE

DISNEY MICKEYMOUSE

DISNEY MICKEYMOUSE

DISNEY MICKEYMOUSE

DISNEY MICKEYMOUSE

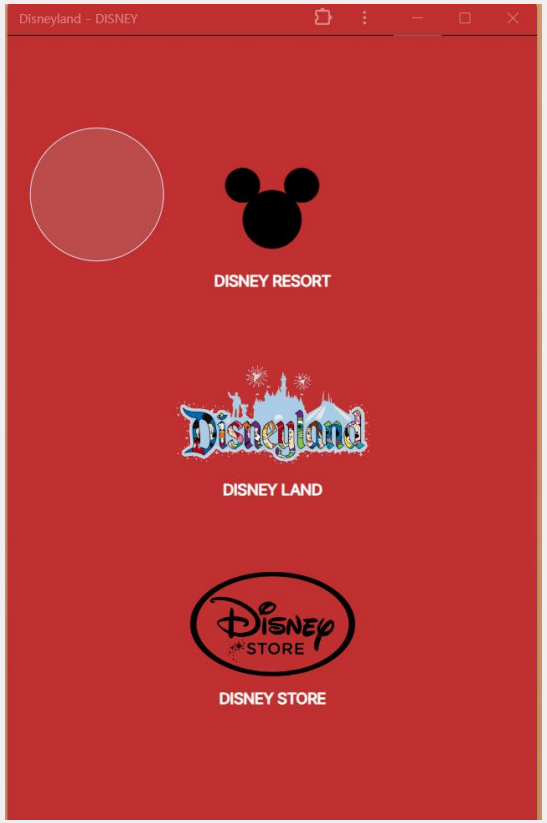
DISNEY MICKEYMOUSE

DISNEY MICKEYMOUSE

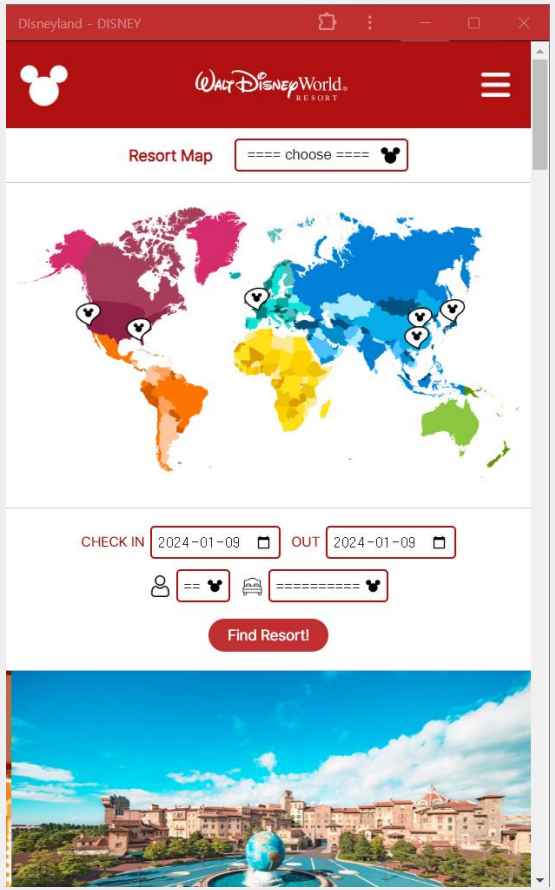
DISNEY MICKEYMOUSE

# UI & UX 앱

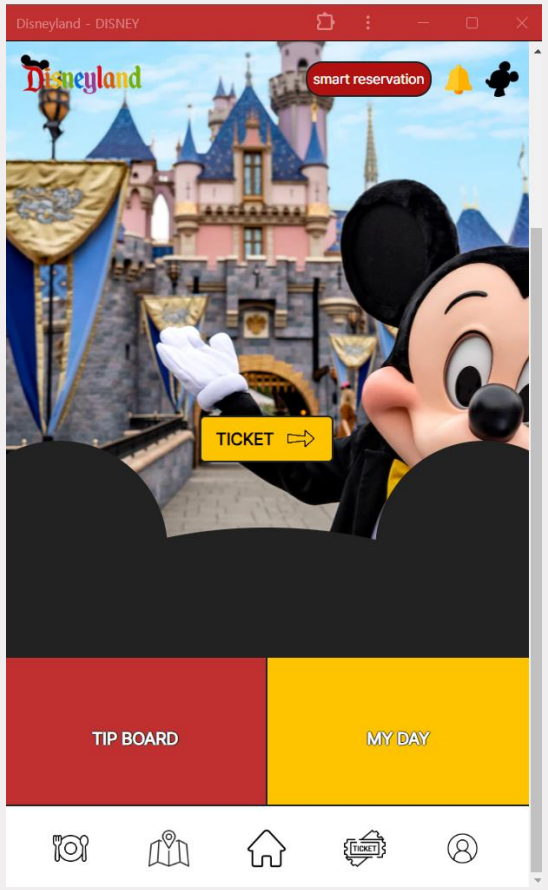
## 메인



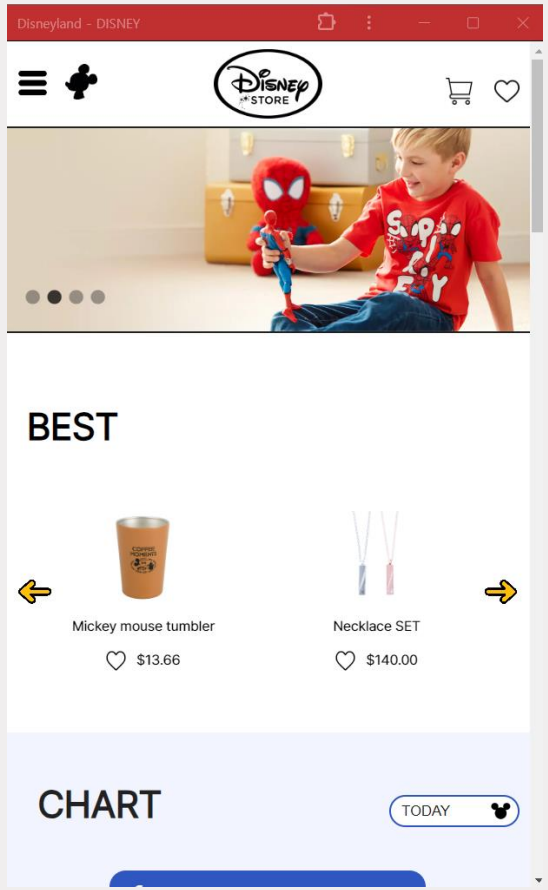

## 디즈니 리조트




## 디즈니 랜드

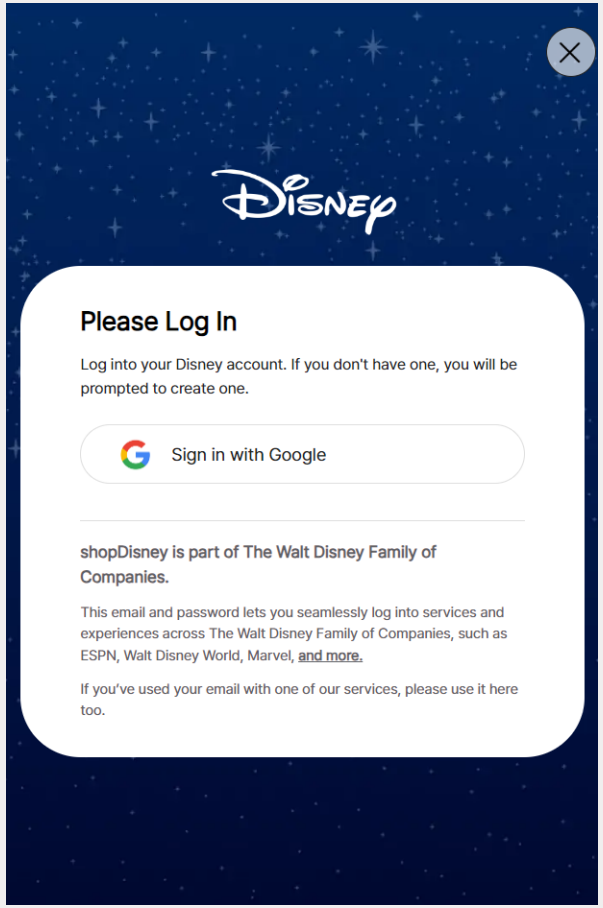



## 디즈니 스토어

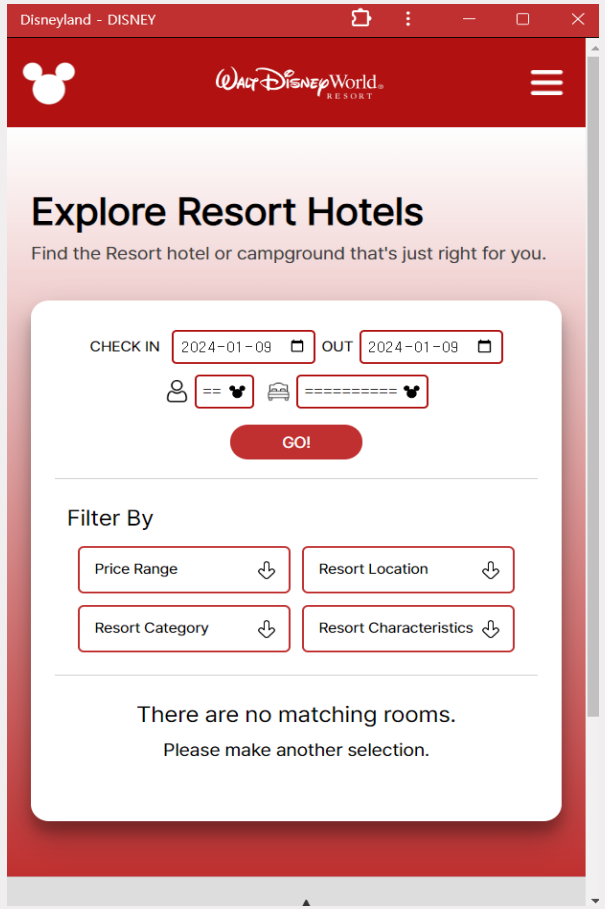


# UI & UX 앱

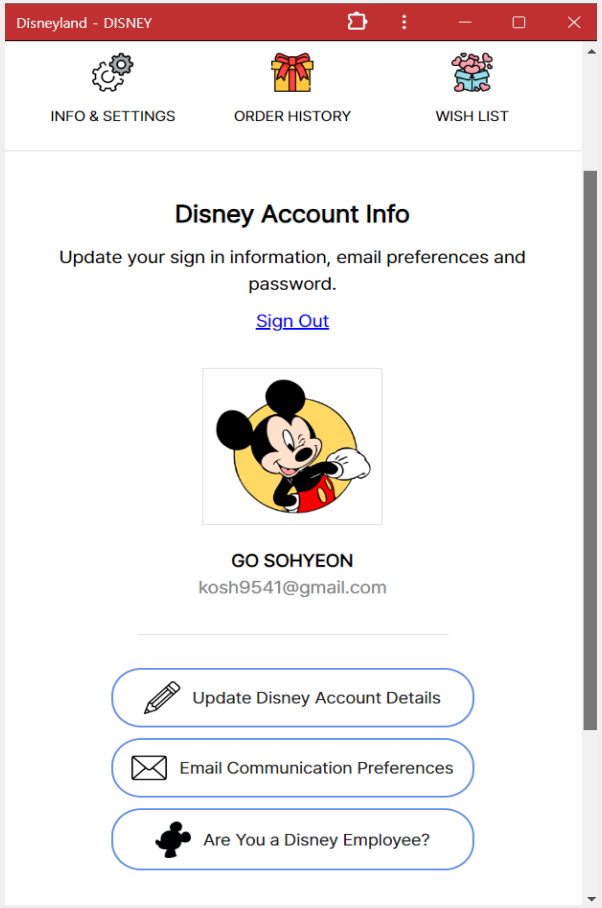
## 로그인 페이지




## 예약 페이지

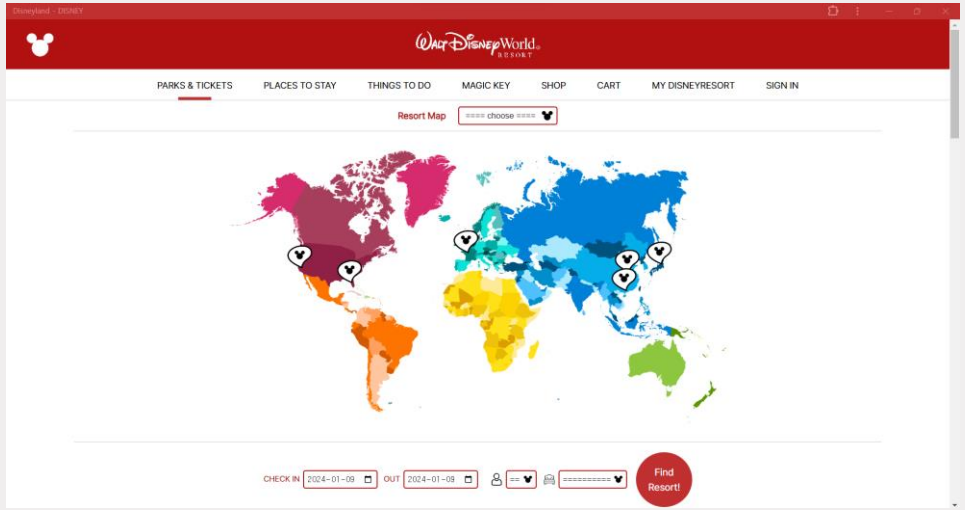



## 회원 페이지

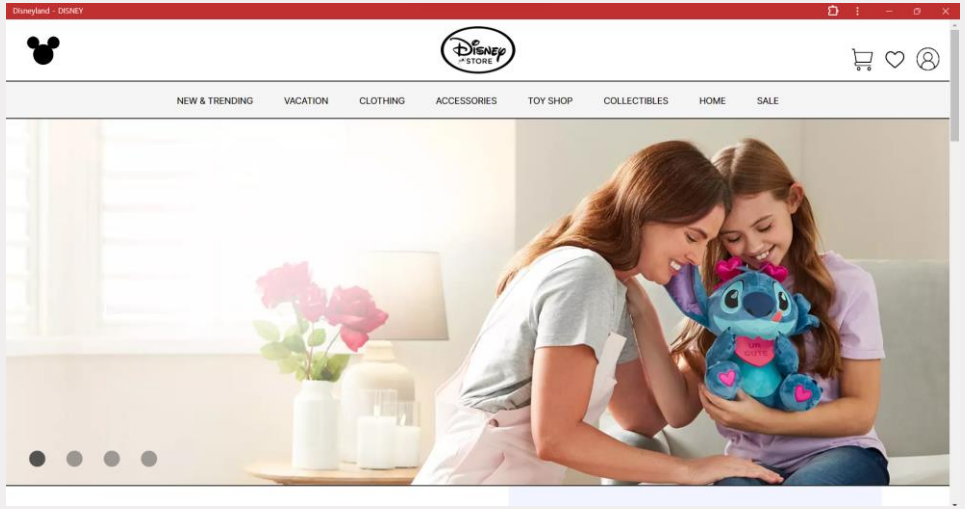


# UI & UX 웹

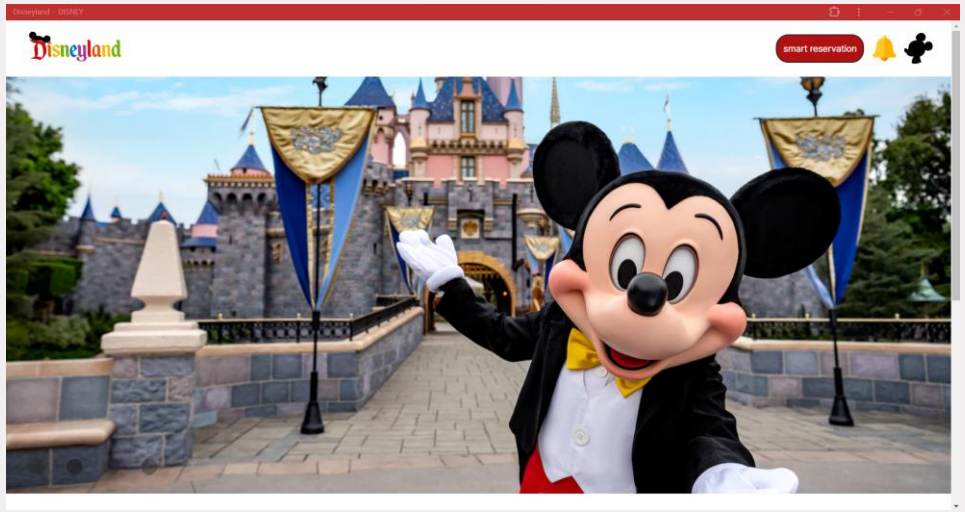
## 디즈니 리조트




## 디즈니 스토어

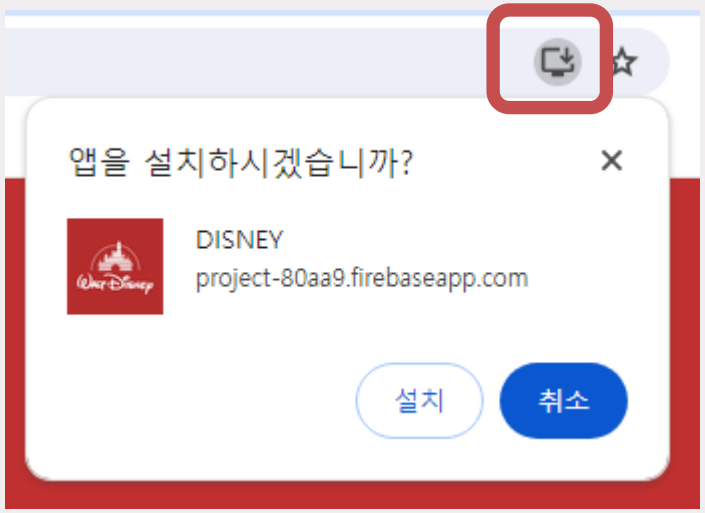


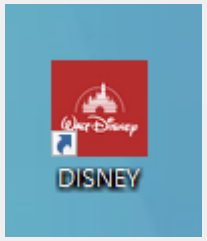

## 디즈니 랜드



# 주요
# 동작 및 기능

# PWA를 이용한 어플 다운


## Manifest.json

manifest.json > ...

```
{
    "name":"Disneyland",
    "shot_name":"DISNEY",
    "description":"Welcome to Disneyland",
    "scope":".",
    "start_url":"./",
    "display":"fullscreen",
    "orientation":"portrait",
    "theme_color":"rgb(192, 48, 48)",
    "background_color":"rgb(192, 48, 48)",
    "icons": [
        {
            "src": "./disney_castle_blue.png",
            "sizes": "192x192",
            "type": "image/png"
        }
    ]
}
```

## Service-worker.js

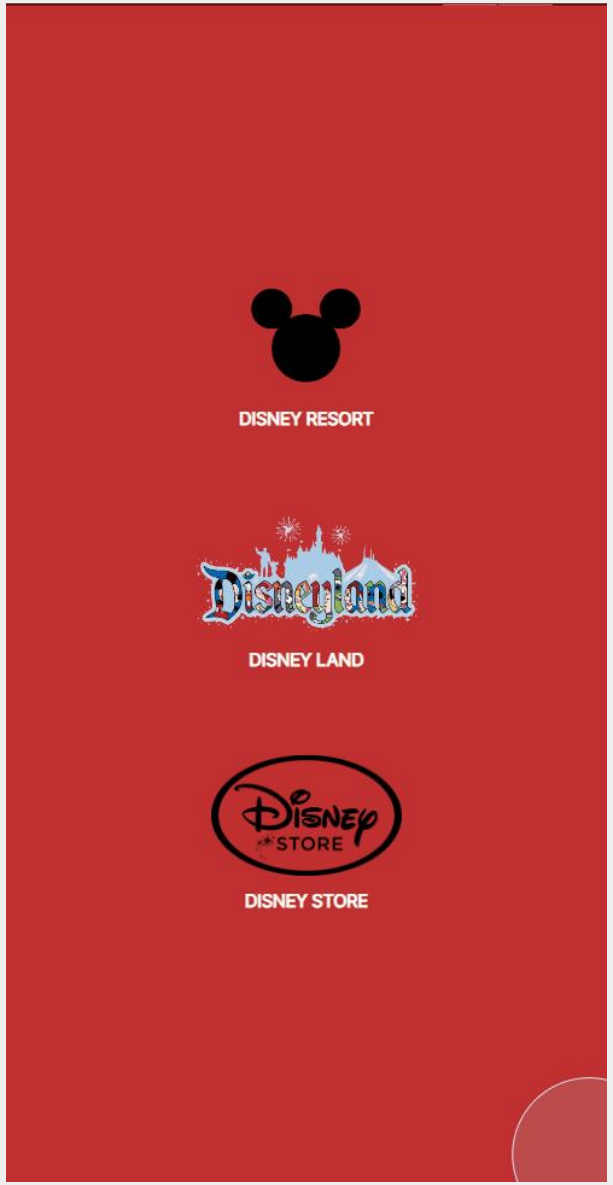
ic > JS service-worker.js > ...

```
// 캐시 제목과 파일 설정
const sCacheName = "disney";      // 캐시 제목
const aFilesToCache = [ // 캐시할 파일 지정
    './',
    './index.html',
    './manifest.json',
    './disney_castle_blue.png'
];

// 서비스워커 실행 & 캐시파일 저장
self.addEventListener("install", pEvent => {
    console.log("서비스 워커 설치 완료!");
    pEvent.waitUntil(
        caches.open(sCacheName)
        .then(pCache => {
            console.log("캐시에 파일 저장 완료!");
            return pCache.addAll(aFilesToCache);
        })
    );
});

// 고유 번호 할당받은 서비스 워커 동작 시작
self.addEventListener('activate', pEvent => {
    console.log('서비스워커 동작 시작됨!');
});

// 고유 번호를 할당받은 서비스워커 작동
self.addEventListener('fetch', pEvent => {
    pEvent.respondWith(
        caches.match(pEvent.request)
        .then(response => {
            if(!response){
                console.log("네트워크로 데이터 요청!", pEvent.request)
                return fetch(pEvent.request)
            }
            console.log("캐시에서 데이터 요청!", pEvent.request)
            return response;
        }).catch(err => console.log(err))
    );
});
```

# 메인 : 마우스 효과

```css
/* 마우스 효과 */

#cursor {
  position: fixed;
  z-index: 100;
  left: 0;
  top: 0;
  pointer-events: none;
  /* will-transform: transform; */
}
1 reference
.cursor__inner {
  width: 8rem;
  height: 8rem;
  border-radius: 50%;
  transform: translate(-50%, -50%);
  border: 1px solid #ece1f5;
  background: #b5878751;
  transition: all .6s cubic-bezier(0.16, 1, 0.3, 1),
  opacity .3s cubic-bezier(0.16, 1, 0.3, 1);
}

0 references | 1 reference
.hide .cursor__inner {
  opacity: 0;
  width: 2.5vw;
  height: 2.5vw;
}

0 references
#cursor.overlay {
  z-index: 1000;
}

0 references | 1 reference
.overlay .cursor__inner {
  width: 3rem;
  height: 3rem;
  background-color: #f28c8599;
  border-color: transparent;
}

3 references
.button::after {
  content: "";
  position: absolute;
  z-index: 0;
  width: 0;
  height: 0;
  border-radius: 50%;
  background-color: #ffffff;
  transition: all .6s cubic-bezier(0.16, 1, 0.3, 1);
}
```

```js
export default {
  name: '',
  components: { disneyResort, disneyLand, disneyStore },
  data() {
    return {
      mouse: { x: -100, y: -100 },
      pos: { x: 0, y: 0 },
      speed: 0.1,
      cursorX: 0,
      cursorY: 0,
    };
  },
  mounted() {
    window.addEventListener("mousemove", this.updateCoordinates);
    this.loop();
    this.initCursor();
  },
  methods: {
    updateCoordinates(e) {
      this.mouse.x = e.clientX;
      this.mouse.y = e.clientY;
    },
    updatePosition() {
      this.pos.x += (this.mouse.x - this.pos.x) * this.speed;
      this.pos.y += (this.mouse.y - this.pos.y) * this.speed;
    },
    loop() {
      this.updatePosition();
      requestAnimationFrame(this.loop);
    },
    initCursor() {
      const cursorModifiers = document.querySelectorAll("[cursor-class]");
      cursorModifiers.forEach((cursorModifier) => {
        cursorModifier.addEventListener("mouseenter", function () {
          let attribute = this.getAttribute("cursor-class");
          cursor.classList.add(attribute);
        });
        cursorModifier.addEventListener("mouseleave", function () {
          let attribute = this.getAttribute("cursor-class");
          cursor.classList.remove(attribute);
        });
      });
    }
  }
};
```



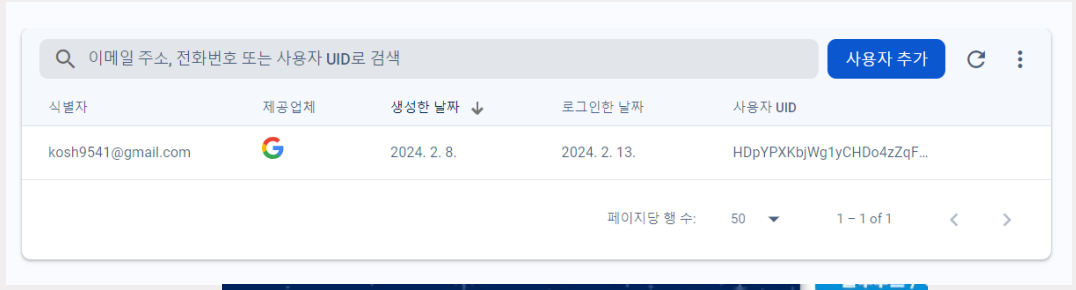
# 로그인 페이지 : 파이어베이스를 활용한 구글 로그인 불러오기

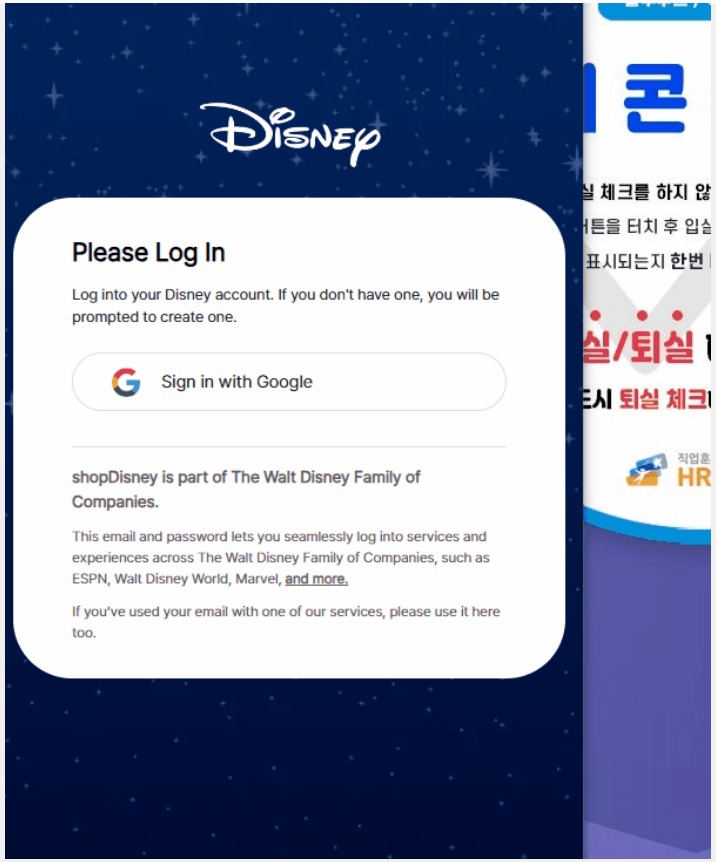
```
methods: {
    async loginWithGoogle() {
        console.log("login");
        const provider = new firebase.auth.GoogleAuthProvider();
        provider.setCustomParameters({
            prompt: "select_account",
        });
        try {
            // signInWithPopup 사용
            const profile = await firebase.auth().signInWithPopup(provider);
            console.log(profile);
            // 로그인 성공 시 disneyStore 페이지로 이동
            this.$router.push('/disneyStore');
        } catch (error) {
            console.error(error);
        }
    },
    onSignIn (googleUser) {
        const profile = googleUser.getBasicProfile();
        console.log('ID Token: ', googleUser.getAuthResponse().id_token); // 실제 토큰
        console.log('ID: ' + profile.getId());
        console.log('Name: ' + profile.getName());
        console.log('Image URL: ' + profile.getImageUrl());
        console.log('Email: ' + profile.getEmail());
    },
    checkUserLogin() {
        // 이전에 로그인되었는지 확인
        const user = firebase.auth().currentUser;
        if (user) {
            // 로그인된 경우, storeMemberPage 페이지로 이동
            this.$router.push('/storeMemberPage');
        }
    },
```

```
created() {
    // 초기에 로그메뉴가 닫혀있도록 설정
    this.menuVisible = false;

    // 사용자가 로그인되어 있는지 확인
    const user = firebase.auth().currentUser;
    if (user) {
        // 로그인된 경우 로그아웃 페이지로 이동
        this.loginLink = '/storeMemberPage';
    }
},
```

```
    logout() {
        firebase.auth().signOut().then(() => {
            // 로그아웃 성공 시 홈페이지로 이동
            this.$router.push('/disneyStore');
        }).catch((error) => {
            console.error(error);
        });
    },
},
created() {
    // Firebase 초기화
    const firebaseConfig = {
        apiKey: "AIzaSyBjX90mHY7NFSgjdhTRdBAZfFMfjbQvBYs",
        authDomain: "project-80aa9.firebaseapp.com",
        projectId: "project-80aa9",
        storageBucket: "project-80aa9.appspot.com",
        messagingSenderId: "400374478942",
        appId: "1:400374478942:web:860606ee80226e78831ccf"
    };
    firebase.initializeApp(firebaseConfig);
    window.onSignIn = this.onSignIn;

    // 로그인 상태 확인
    this.checkUserLogin();
},
```

```
methods: {
    async signOut() {
        try {
            await firebase.auth().signOut();
            // 로그아웃 후 disneyStore 페이지로 이동
            this.$router.push('/disneyStore');
        } catch (error) {
            console.error(error);
        }
    }
}
```



# 디즈니 리조트 : 옵션 선택에 따른 구글 지도 API 불러오기


```
<!-- select box -->
<div class="choose_resort">
    <form method="get" action="#">
        <label for="Map">Resort Map</label>
        <select v-model="selectedResort" @change="updateBanner" name="resort map" id="Map" class="select_1">
            <option value="none">==== choose ====</option>
            <option value="California">California, USA</option>
            <option value="Orlando">Orlando, USA</option>
            <option value="Paris">Paris, France</option>
            <option value="Hongkong">Hongkong</option>
            <option value="Shanghai">Shanghai, China</option>
            <option value="Tokyo">Tokyo, Japan</option>
        </select>
    </form>
</div>
<!-- banner -->
<div class="main_banner">
    <div v-if="selectedResort === 'none'">
        <img :src="bannerImage" alt="">
    </div>
    <div v-else>
        <div ref="map" class="map_w"></div>
    </div>
</div>
```

# 디즈니 리조트 : 옵션 선택에 따른 구글 지도 API 불러오기

```
data() {
    return {
        selectedResort: 'none',
        selectedPersons: 'none',
        selectedRoom: 'none',
        bannerImage: require('../assets/images/map.png'), // 이미지의 경로로 설정
        mapInitialized: false // 구글 지도 초기화 여부를 추적하기 위한 변수 추가
    };
},
methods: {
    updateBanner() {
        if (this.selectedResort !== 'none') {
            this.loadGoogleMap();
        } else {
            this.bannerImage = require('../assets/images/map.png');
        }
    },
    loadGoogleMap() {
        if (!window.google) {
            const script = document.createElement('script');
            script.src = `https://maps.googleapis.com/maps/api/js?key=AIzaSyBxNhKF7Hc7cbBhbwIH1pnoZkjJamb4t4I&callback=initMap&language=en`;
            script.defer = true;
            script.async = true;
            document.body.appendChild(script);
        } else {
            this.initMap();
        }
    },
```

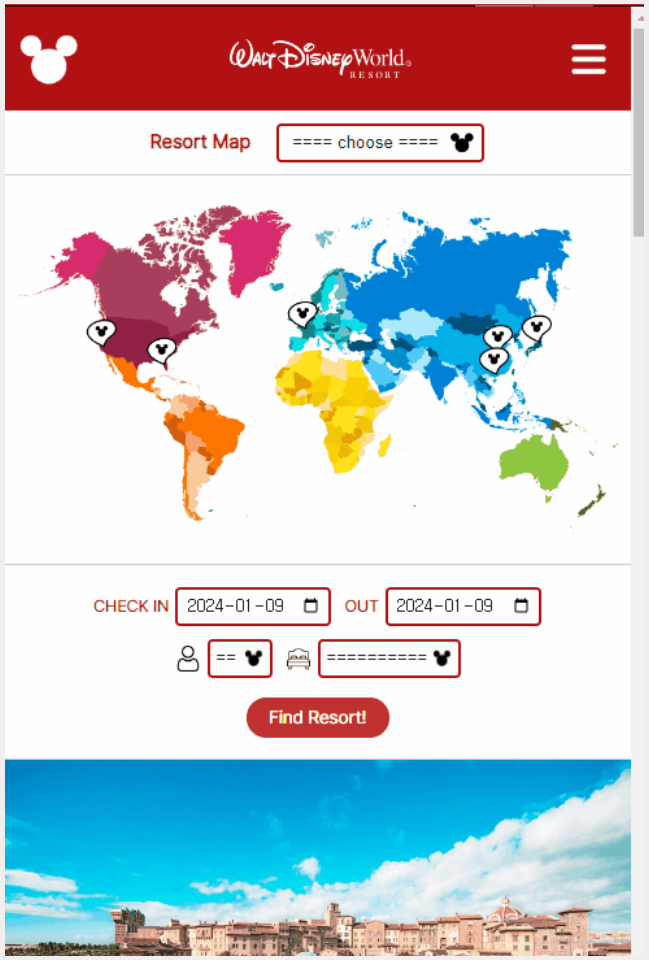
```
    const mapElement = this.$refs.map;
    if (mapElement && google && google.maps) { // Google Maps API가 정의되었는지 확인
        const mapOptions = {
            center: centerLatLng,
            zoom: 15
        };
        const map = new google.maps.Map(mapElement, mapOptions); // map 객체를 생성
        this.mapInitialized = true; // 지도가 초기화되었음을 표시
        console.log(map); // map 변수를 콘솔에 로그
    } else {
        console.error('Google Maps API is not loaded or mapElement is not defined.'); // 오류 메시지를 출력
    }
```

```
initMap() {
    let centerLatLng;
    switch (this.selectedResort) {
        case 'California':
            centerLatLng = { lat: 33.81126022338867, lng: -117.92205810546875 };
            break;
        case 'Orlando':
            centerLatLng = { lat: 28.418790817260742, lng: -81.58479309082031 };
            break;
        case 'Paris':
            centerLatLng = { lat: 48.87063217163086, lng: 2.779606580734253 };
            break;
        case 'Hongkong':
            centerLatLng = { lat: 22.31024932861328, lng: 114.04550170898438 };
            break;
        case 'Shanghai':
            centerLatLng = { lat: 31.142351150512695, lng: 121.66680908203125 };
            break;
        case 'Tokyo':
            centerLatLng = { lat: 35.63071823120117, lng: 139.88287353515625 };
            break;
        default:
            centerLatLng = { lat: 0, lng: 0 };
            break;
    }
```

```
updated() {
    if (this.selectedResort !== 'none' && !this.mapInitialized) {
        // 선택한 리조트가 있고 지도가 초기화되지 않은 경우, 지도를 초기화
        this.initMap();
    }
}
```

```
watch: {
    selectedResort(newResort) {
        if (newResort === 'choose') {
            this.updateBanner();
        }
    },
},
```

# 디즈니 리조트 : 예약 옵션 미 선택시 팝업창 뜨기

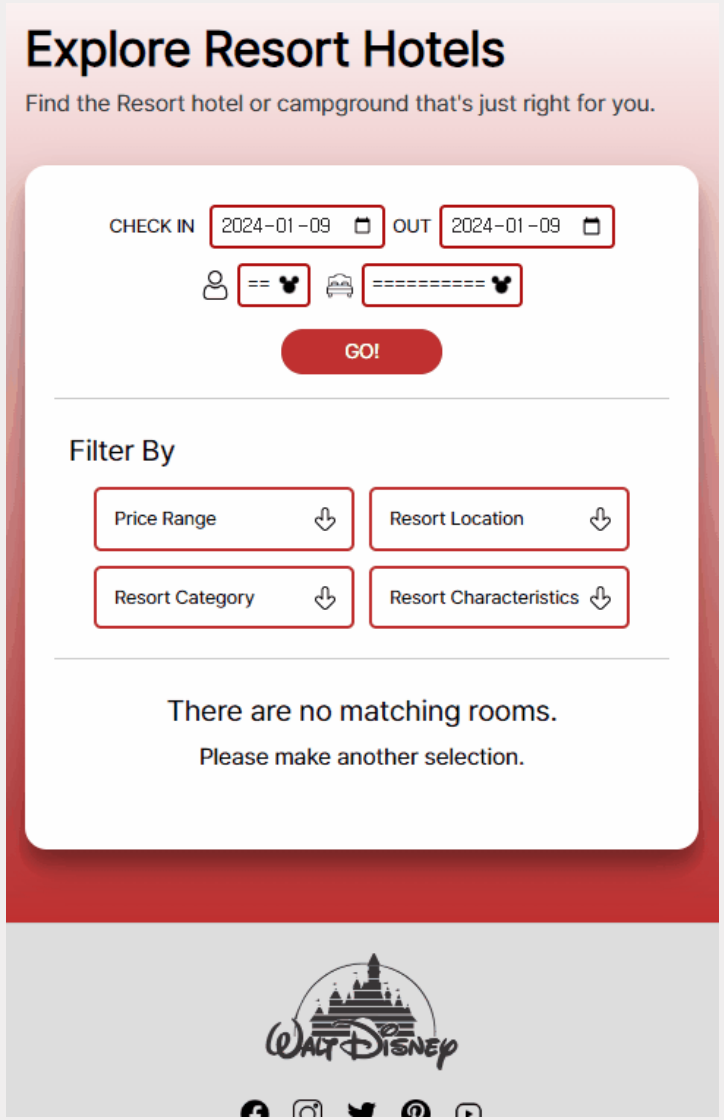



```vue
<!-- 예약 기능 -->
<div class="re_wrap">
    <div class="reservation_wrap">
        <div class="date_wrap">
            <label for="inDate" class="date">
                <p>CHECK IN </p>
                <input type="date" id="inDate" max="2077-07-17" value="2024-01-09">
            </label>
            <label for="outDate" class="date">
                <p>OUT</p>
                <input type="date" id="outDate" max="2077-07-17" value="2024-01-09">
            </label>
        </div>
        <div class="people_room">
            <form action="#">
                <label for="sub">
                    <img src="@/assets/images/people.png" alt="">
                </label>
                <select v-model="selectedPersons" name="persons" size="1" class="select_2">
                    <option value="none" selected>==</option>
                    <option value="1">1</option>
                    <option value="2">2</option>
                    <option value="3">3</option>
                    <option value="4">4</option>
                </select>
            </form>
```

```vue
            <form action="#">
                <label for="room">
                    <img src="@/assets/images/room.png" alt="">
                </label>
                <select v-model="selectedRoom" name="room" id="room" class="select_3">
                    <option value="none" selected>==========</option>
                    <option value="luxury type">luxury type</option>
                    <option value="Deluxe type">Deluxe type</option>
                    <option value="Moderate type">Moderate type</option>
                    <option value="value type">value type</option>
                </select>
            </form>
        </div>
    </div>
    <router-link to="#" class="check_wrap" @click="checkAndProceed">
        <p>Find Resort!</p>
    </router-link>
</div>
```

```vue
    checkAndProceed() {
        if (this.selectedPersons === 'none' || this.selectedRoom === 'none') {
            alert('Please select all options.');
        } else {
            this.$router.push('/resortTicket');
        }
    },
},
```

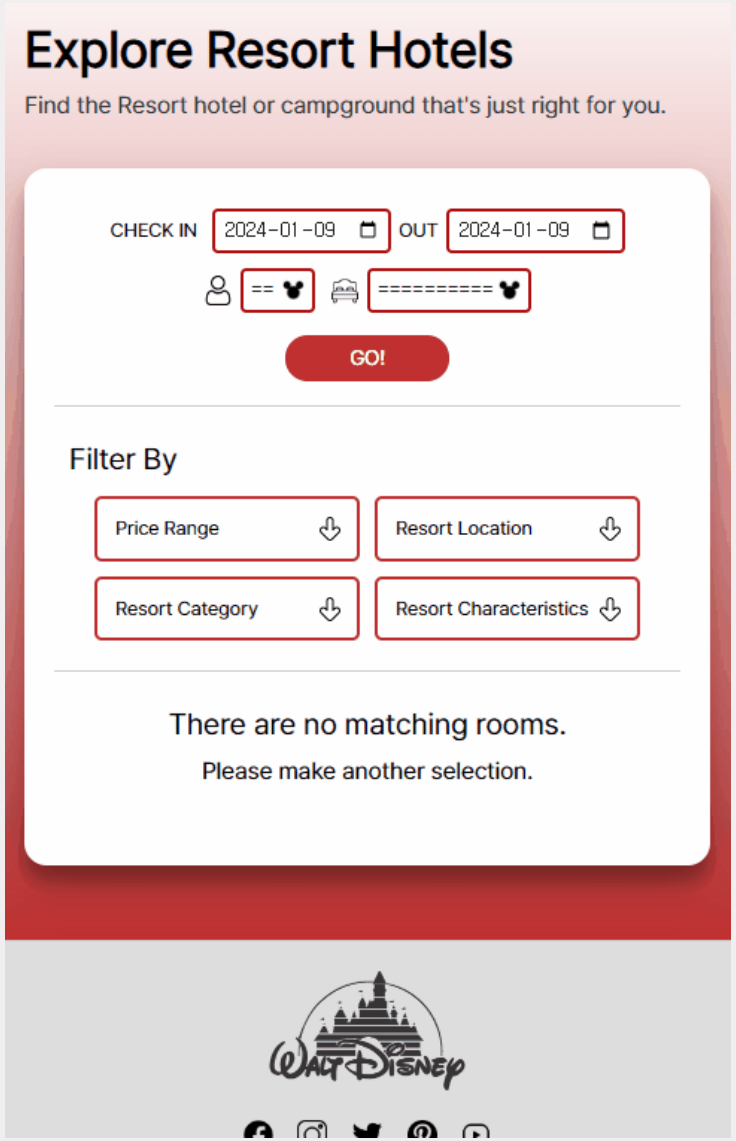
# 디즈니 리조트 : 선택된 옵션 초기화 하기




```
computed: {
    selectedItems() {
        const selectedList = this.chartLists.find(list => list.option === this.selectedOption);
        return selectedList ? selectedList.items : [];
    }
},
```

```
clearFilters(index) {
    // 선택된 항목 초기화
    this.selectedItems[index] = Array(this.filterCategories[index].items.length).fill(false);
},
checkAndProceed() {

},
clearAllFilters() {
    // 모든 선택된 항목 초기화
    this.selectedItems = this.filterCategories.map(() => Array(this.filterCategories.length).fill(false));
    this.visibleCategories = [];
    this.activeCategory = null;
},
handleInputSelection() {
    // input 선택을 처리하는 메서드
    this.isInputSelected = true;
},
clearFilters(index) {
    // 선택된 항목 초기화
    this.selectedItems[index] = Array(this.filterCategories[index].items.length).fill(false);
    // input 선택 여부 초기화
    this.isInputSelected = this.hasSelectedItems;
},
clearAllFilters() {
    // 모든 선택된 항목 초기화
    this.selectedItems = this.filterCategories.map(() => Array(this.filterCategories.length).fill(false));
    this.visibleCategories = [];
    this.activeCategory = null;
    // 선택된 항목이 없으므로 isInputSelected를 false로 설정
    this.isInputSelected = false;
},
```

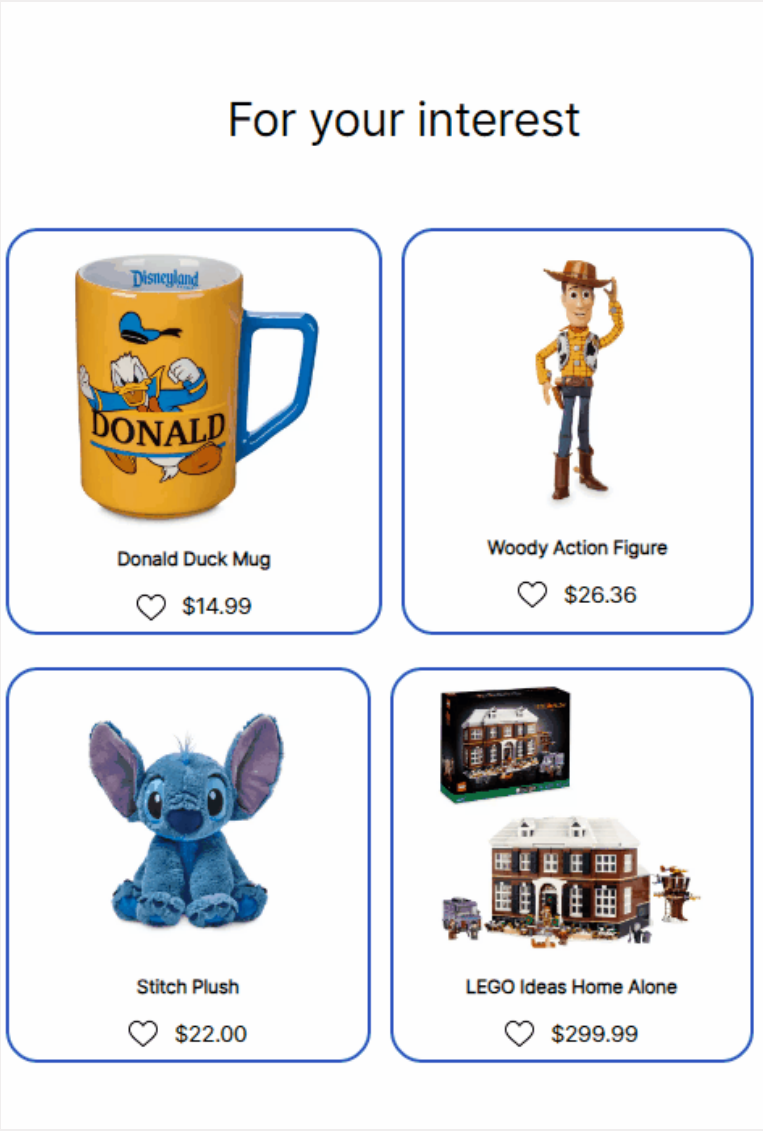
# 디즈니 리조트 : 예약 옵션 선택시 해당 목록 나오기




```vue
<!-- 가능한 호텔 목록 -->
<section class="filter_wrap">
    <form @submit.prevent="handleSubmit">
        <div class="filter_by">
            <p>Filter By</p>
        </div>
        <div class="fieldset_wrap">
            <fieldset v-for="(category, index) in filterCategories" :key="index">
                <div class="arrow_wrap" @click="toggleOlWrap(index)" :class="{ 'active': visibleCategories.includes(index) }">
                    <legend>
                        <span>{{ category.name }}</span>
                    </legend>
                </div>
                <div :class="{ 'ol_wrap': true, 'visible': visibleCategories.includes(index) }">
                    <ol>
                        <li v-for="(item, itemIndex) in category.items" :key="itemIndex">
                            <input type="checkbox" :id="`${index}-${itemIndex}`" :name="`${index}-${itemIndex}`" v-model="selectedItems[
                            <label :for="`${index}-${itemIndex}`">
                                <span class="icon"></span>
                                <span class="value">{{ item }}</span>
                            </label>
                        </li>
                    </ol>
                    <div class="clear_wrap">
                        <span @click="clearFilters(index)">Clear Filters</span>
                    </div>
                </div>
            </fieldset>
        </div>
    </form>
    <div class="selected_wrap" v-show="hasSelectedItems">
        <ul>
            <li>
                <span v-for="(category, categoryIndex) in filterCategories" :key="categoryIndex">
                    <span v-for="(item, itemIndex) in category.items" :key="itemIndex" v-show="selectedItems[categoryIndex][itemIndex]">
                        {{ item }},
                    </span>
                </span>
            </li>
        </ul>
        <div class="clearAll_wrap">
            <span @click="clearAllFilters">Clear All</span>
        </div>
    </div>
</section>
```

```vue
adjustFieldsetWrapPosition(index) {
    this.$nextTick(() => {
        const olWrap = document.querySelector(`.fieldset_wrap:nth-child(${index + 1}) .ol_wrap`);
        if (olWrap) {
            const olWrapHeight = olWrap.offsetHeight;
            this.fieldsetWrapTop = `${olWrapHeight}px`;
        }
    });
},
```

# 디즈니 스토어 : 하트 찜 버튼 구현




```
toggleLike(index, isRecommended) {
    if (isRecommended) {
        if (this.recommendedItems[index]) {
            this.recommendedItems[index].isLiked = !this.recommendedItems[index].isLiked;
        }
    } else {
        if (this.bestItems[index]) {
            this.bestItems[index].isLiked = !this.bestItems[index].isLiked;
        }
    }
},
```

# 디즈니 스토어 : 차트 옵션 선택 시 해당 차트 불러오기


```
computed: {
    selectedItems() {
        const selectedList = this.chartLists.find(list => list.option === this.selectedOption);
        return selectedList ? selectedList.items : [];
    }
},
```

THANK YOU

# 프로젝트 바로가기