카뮤니티 발표자료

김현수, 조현우

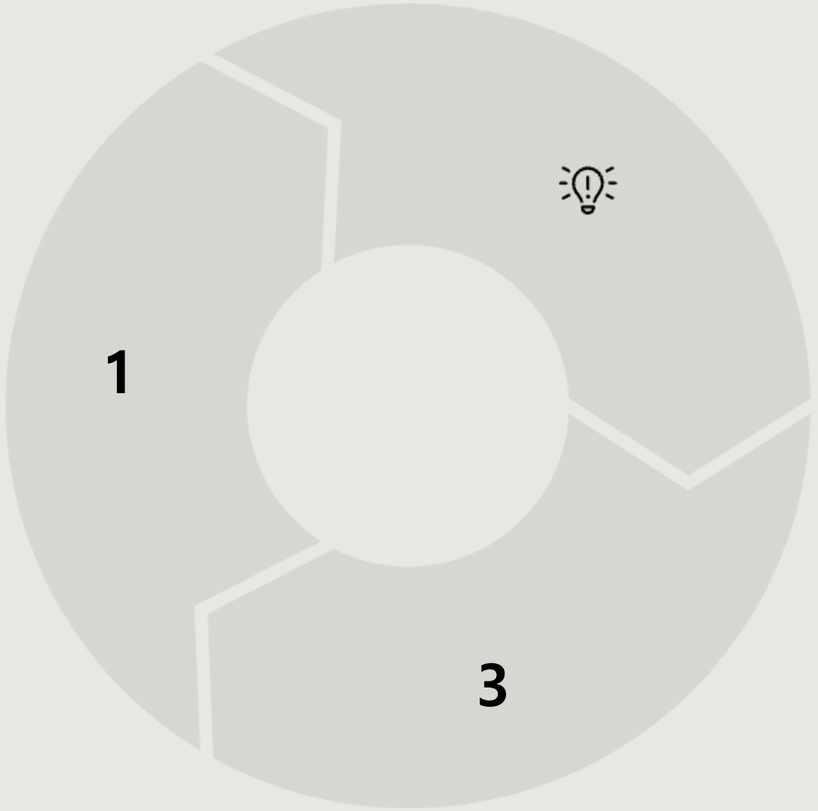
목차

주제 선정 과정

왜 (Why)

튜닝시장 규모의 지속적 증가

차량에 대한 인식 변화



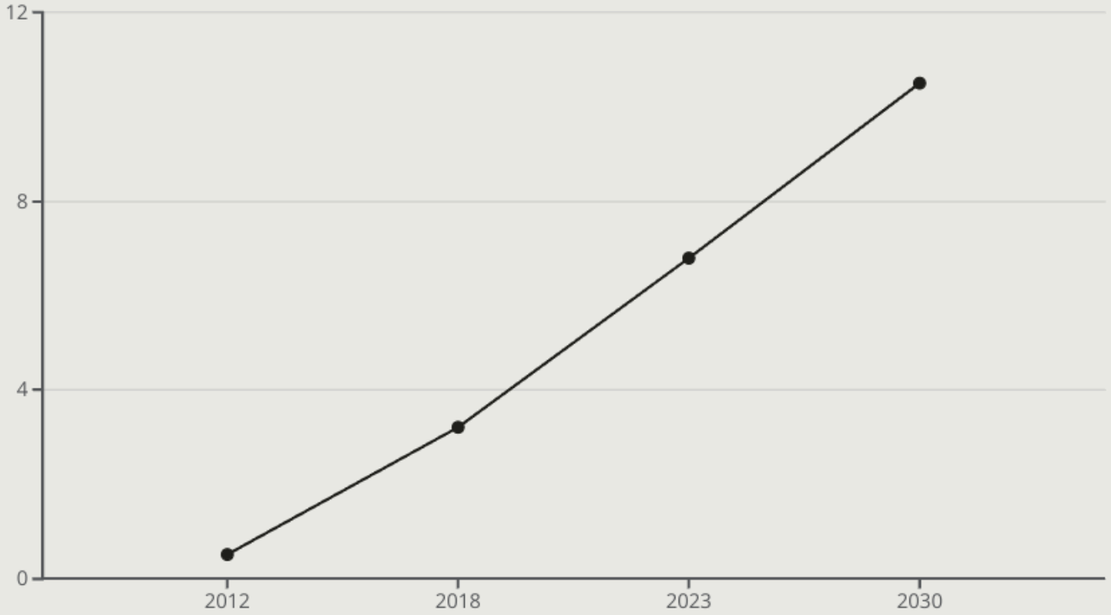
무엇을 (What)

차량 튜닝 결과 확인 가능 서비스의 부재

어떻게 (How)

튜닝 시뮬레이션으로 결과 예측 및 비용 절감

왜 차량 튜닝 서비스인가?



튜닝 시장의 빠른 성장

2012년 0.5조원 → 2030년 10.5조원 예상

맞춤형 소비 수요

차량 소유자 68.2%, 개성 표현 위해 튜닝 고려

인증부품 증가

213,045건 (2023년), 전년 대비 16.5% 증가

정부 규제 완화

튜닝 승인 간소화, 전담창구 확대 시행

출처: 국토교통부 자동차튜닝산업 실태조사(2023), 자동차튜닝협회 인증부품 통계(2024)

무엇이 문제인가?

튜닝 시도

시뮬레이터 없음

커뮤니티 의존

비용·시간 낭비

높은 비용과 반품·버림

튜닝 결과가 어떤지 모르고 사서 달아봤다가 '**별로**'면 **교환·반품** 혹은 **폐기**해야 해서 **비용적으로 부담** 큼.

시행착오 반복 & 정보 신뢰도 저하

커뮤니티 후기와 블로그 사례를 참고하지만, 차량·부품 조합이 다 달라서 결국 **직접 시도**할 수밖에 없음. 이 과정에서 **시간·피로감 증가**.

어떻게 문제를 해결할것인가

차량 선택

파츠 조합

튜닝 결과

비용 절감

핵심 이점

시뮬레이션 피드백

시각적 결과 즉시 확인

다양한 조합

무제한 가상 테스트

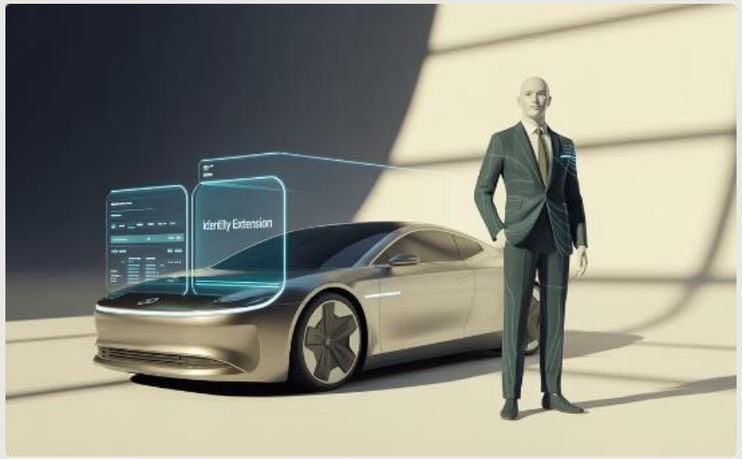
비용 절감

불필요한 지출 감소

프로젝트 슬로건

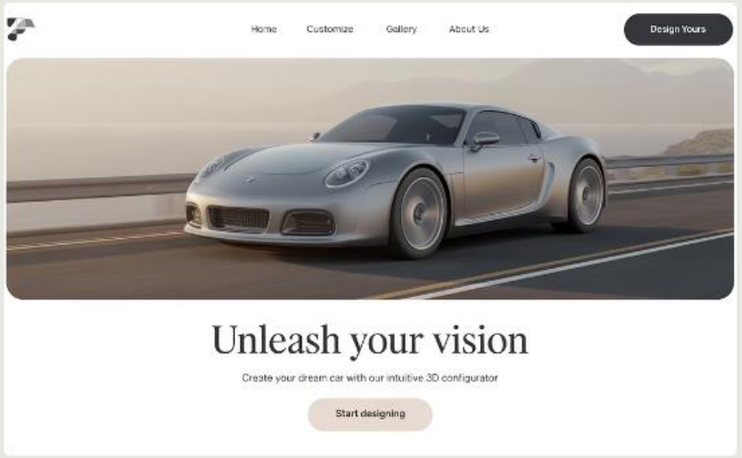
패션처럼 선택하는 튜닝

옷을 고르듯 차량 부품을 선택합니다.



디지털 자아 확장

차량은 개인 정체성의 연장선입니다.



라이프스타일 경험

튜닝은 기계적 작업이 아닌 생활 방식입니다.

사용한 기술 스택

언어

Java

프레임워크

Spring Boot, Spring Data JPA, Spring Security

데이터베이스

MySQL

개발 도구

IntelliJ, MySQL Workbench

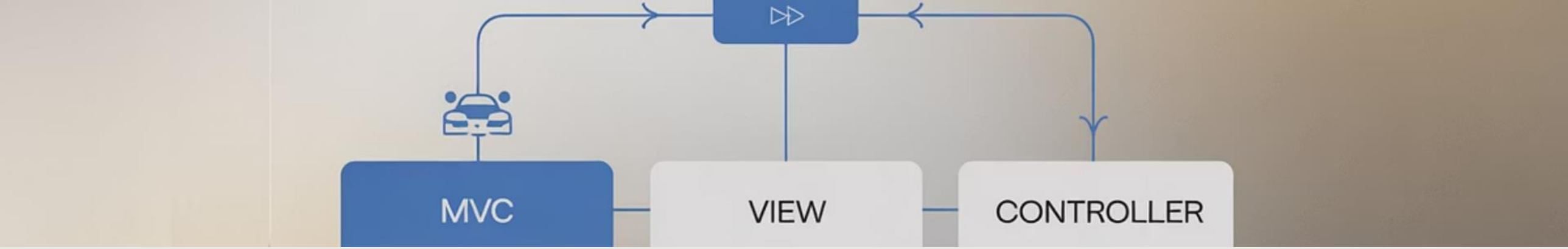
협업 도구

GitHub

테스트 도구

Postman

Architecture Overview



프로젝트 구조 및 흐름

View

사용자 인터페이스

AJAX 요청/응답

비동기 통신

Controller

요청 처리 및 응답

Service

비즈니스 로직 처리

Model

데이터 관리

세부 흐름 구조

프로젝트의 계층별 구조와 사용 기술

계층	역할 설명	사용 기술 or 클래스 예시
View	사용자 인터페이스 계층	HTML + JS
Config	설정 및 보안 처리	WebSecurityConfig, SecurityConfig
Controller	요청 매핑 및 처리	UserController, PostController 등
Service	비즈니스 로직 처리	UserService, TuningService 등
Repository	데이터 접근 계층	UserRepository등
Model	Entity, DTO 구성	User, Car, TuningPart, PostDTO 등

보안 기술 세부사항

이메일 인증

사용자 신원 확인

JWT 인증

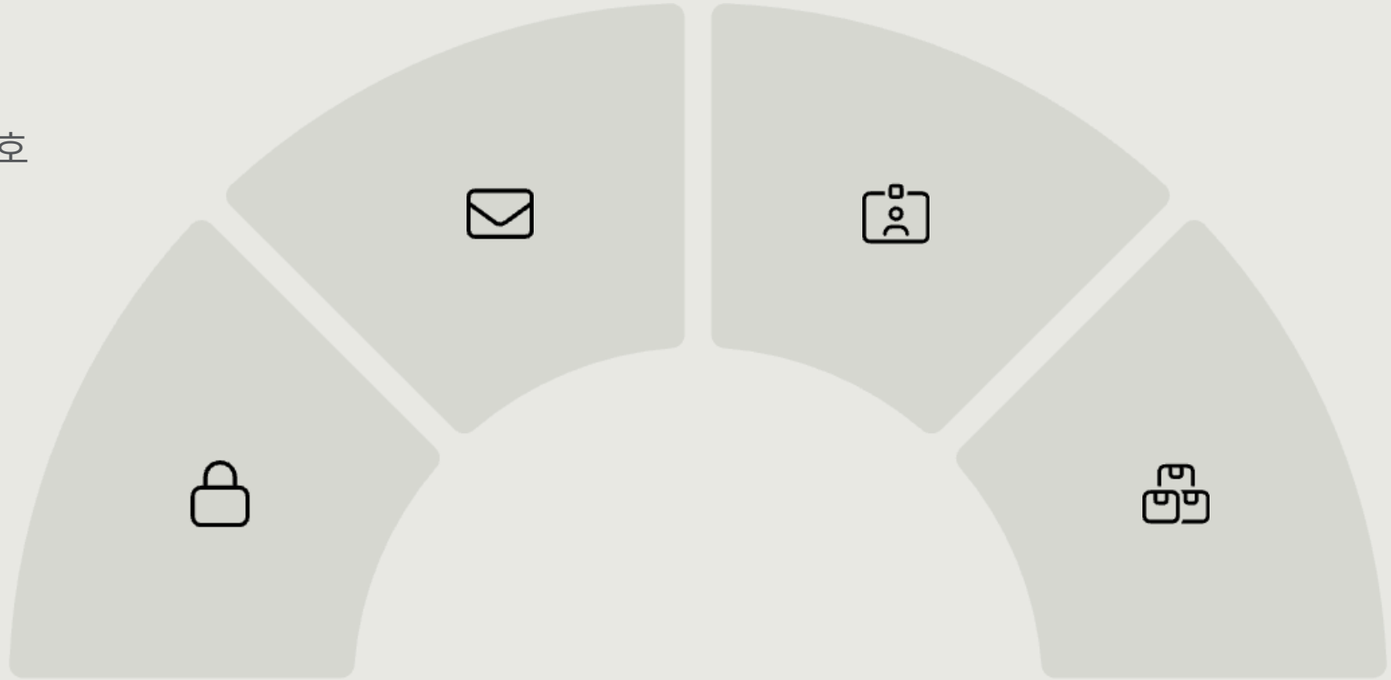
안전한 토큰 기반 인증

비밀번호 암호화

안전한 사용자 정보 보호

DTO 최소 전달

필요한 데이터만 전송



비밀번호 암호화 과정

비밀번호 암호화 코드

```java
// 비밀번호 암호화를 위한 인터페이스
@Bean  Sungwon Lee *
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
// 회원 가입 코드 일부
User user = User.builder()
        .userId(request.getUserId())
        .nickname(request.getNickname())
        .password(passwordEncoder.encode(request.getPassword()))
        .email(request.getEmail())
        .phone(request.getPhone())
        .name(request.getName())
        .introduction("자기소개가 아직 없습니다.")
        .profileImagePath("/image/UserImageDefault.png")
        .build();
```

인코딩 전후 비교

구분	예시 값
입력 비밀번호	mySecret123!
암호화된 비밀번호	$2a$10$W3bY7hV5Fx9KjkDf0...

보안 기술 세부사항 - 이메일 인증

이메일 전송 코드

```java
@Component  Sungwon Lee *
@RequiredArgsConstructor
public class SmtpEmailSender implements EmailSender {   // 이메일 전송을 위한 클래스

    private final JavaMailSender mailSender;

    @Override  2 usages  Sungwon Lee *
    public void send(String to, String subject, String text) {
        MimeMessage message = mailSender.createMimeMessage();
        try {
            MimeMessageHelper helper = new MimeMessageHelper(message,  multipart: false,  encoding: "UTF-8");
            helper.setTo(to);
            helper.setSubject(subject);
            helper.setText(text,  html: false);
            helper.setFrom("brian7536@gmail.com");
            mailSender.send(message);
        } catch (MessagingException e) {
            throw new RuntimeException("이메일 전송 실패: " + e.getMessage());
        }
    }
}
```

이메일 인증 코드

```java
    // 이메일 인증 코드 검증
    emailVerificationService.verifyAndMarkAsVerified(
            request.getEmail(),
            request.getVerificationCode(),
             purpose: "signup"
    );
```

🚫 비인가 접근 차단

무단 접근 방지

🔐 계정 소유자 확인

이메일을 통한 본인인증

보안 기술 세부사항 - DTO

```java
@Getter  3 usages  Sungwon Lee *
@AllArgsConstructor
public class UserResponse {
    private String nickname;
    private String email;
    private String phone;
    private String name;
    private LocalDateTime createdAt;
    private String introduction;
    private String profileImagePath;
}
```

DTO 사용 목적

- 필요한 정보만 주고 받음

보안 이점

- 민감 정보 필터링

보안 세부사항-JWT 토큰

JWT 인증 흐름:

1. 로그인 시 JWT 생성
2. 클라이언트에 토큰 저장
3. 요청 시 Header에 포함
4. 서버에서 토큰 검증

사용자 권한 확인

토큰 기반 검증으로 안전한 접근 제어

서버 부하 감소

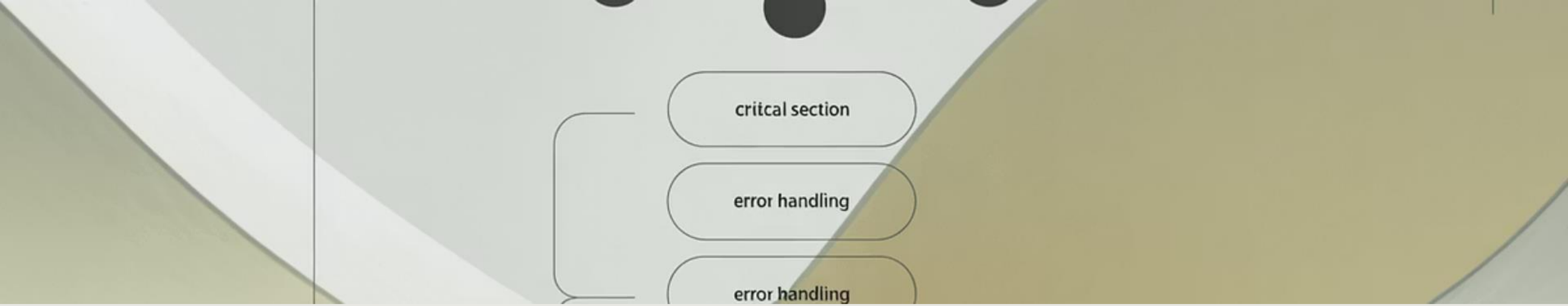
세션 저장소 불필요

로그인 성공 -> jwt token 발급

```java
// 로그인 처리
public UserLoginResponse login(UserLoginRequest request) {  1 usage   Sungwon Lee *
    try {
        authenticationManager.authenticate(
                new UsernamePasswordAuthenticationToken(request.getUserId(), request.getPassword())
        );
    } catch (Exception e) {
        throw new IllegalArgumentException("아이디 또는 비밀번호가 올바르지 않습니다.");
    }

    // jwt token 발급
    String token = jwtTokenProvider.createToken(request.getUserId());
    return new UserLoginResponse(token);
}
```

Jwt 토큰 생성 코드

```java
// JWT 토큰 생성
public String createToken(String userId) {  1 usage   Sungwon Lee *
    Date now = new Date();
    Date expiry = new Date(now.getTime() + expiration);

    return Jwts.builder()
            .setSubject(userId)
            .setIssuedAt(now)
            .setExpiration(expiry)  // 유효 시간 설정
            .signWith(key, SignatureAlgorithm.HS256)
            .compact();
}
```



기타 기술 세부사항

락 처리 및 동시성 제어

다중 사용자 환경에서 데이터 일관성 유지

공통 예외 처리 적용

일관된 오류 응답 및 처리

Lombok 사용으로 코드 간결화

반복적인 코드 제거로 개발 효율성 향상

기타 기술 세부사항 - 동시성 제어

중복 처리 방지

여러 사용자의 좋아요 요청에 대한 동시성 문제를 해결합니다.

데이터 무결성 보장

락 메커니즘을 활용해 일관성을 유지합니다.

```
@Lock(LockModeType.PESSIMISTIC_WRITE)  1 usage  new *
@Query("SELECT c FROM Comment c WHERE c.commentId = :commentId")
Optional<Comment> findByIdForUpdate(@Param("commentId") Integer commentId);
```

기타 기술 세부사항 - 공통 예외처리

🧭 예외 흐름 일원화

서비스 전체의 예외를 **미리 정의된 구조로 처리합니다**.

📦 유지보수 효율성 향상

컨트롤러에서 발생한 모든 예외는 공통 예외 핸들러를 통해 처리되 일관된 응답 형식으로 전달합니다.

공통 에러 코드

```
public enum ErrorCode {  no usages

    // 공통 에러
    INVALID_INPUT_VALUE(HttpStatus.BAD_REQUEST,  message: "입력 값이 올바르지 않습니다."),  no usages
    ENTITY_NOT_FOUND(HttpStatus.NOT_FOUND,  message: "존재하지 않는 리소스입니다."),  no usages
    DUPLICATE_RESOURCE(HttpStatus.CONFLICT,  message: "이미 존재하는 리소스입니다."),  no usages
    UNAUTHORIZED(HttpStatus.UNAUTHORIZED,  message: "인증이 필요합니다."),  no usages
    FORBIDDEN(HttpStatus.FORBIDDEN,  message: "권한이 없습니다."),  no usages
    INTERNAL_SERVER_ERROR(HttpStatus.INTERNAL_SERVER_ERROR,  message: "서버 내부 오류입니다."),  no usages

    // 게시글 좋아요 관련 에러
    POST_ALREADY_LIKED(HttpStatus.CONFLICT,  message: "이미 좋아요를 누른 게시글입니다."),  no usages
    POST_LIKE_NOT_FOUND(HttpStatus.NOT_FOUND,  message: "좋아요한 게시글을 찾을 수 없습니다."),  no usages

    // 댓글 좋아요 관련 에러
    COMMENT_ALREADY_LIKED(HttpStatus.CONFLICT,  message: "이미 좋아요를 누른 댓글입니다."),  no usages
    COMMENT_LIKE_NOT_FOUND(HttpStatus.NOT_FOUND,  message: "좋아요한 댓글을 찾을 수 없습니다.");  no usages
```

CustomException 코드

```
public class CustomException extends RuntimeException {  no usages

    private final ErrorCode errorCode;  2 usages

    public CustomException(ErrorCode errorCode) {  no usages
        super(errorCode.getMessage());
        this.errorCode = errorCode;
    }

    public ErrorCode getErrorCode() {  no usages
        return errorCode;
    }
}
```

기타 기술 세부사항 - 공통 예외처리

전역 예외 처리 코드

```java
@RestControllerAdvice  Sungwon Lee *
public class GlobalExceptionHandler {

    @ExceptionHandler(CustomException.class)  new *
    public ResponseEntity<ErrorResponse> handleCustomException(CustomException e) {
        ErrorCode errorCode = e.getErrorCode();
        ErrorResponse response = new ErrorResponse(errorCode.getStatus().value(), errorCode.getMessage());
        return new ResponseEntity<>(response, errorCode.getStatus());
    }

    public static class ErrorResponse {  3 usages  Sungwon Lee *
        private final int status;  2 usages
        private final String message;  2 usages

        public ErrorResponse(int status, String message) {  1 usage  new *
            this.status = status;
            this.message = message;
        }

        public int getStatus() {  new *
            return status;
        }

        public String getMessage() {  new *
            return message;
        }
    }
```

롬복으로 코드 품질 향상하기

✂️ 반복 코드 제거

롬복을 통해 반복되는 getter, setter, 생성자와 같은 코드를 간결하게 표현합니다.

👓 가독성 향상

어노테이션을 통해 간결하게 표현하여 가독성을 향상시켰습니다.

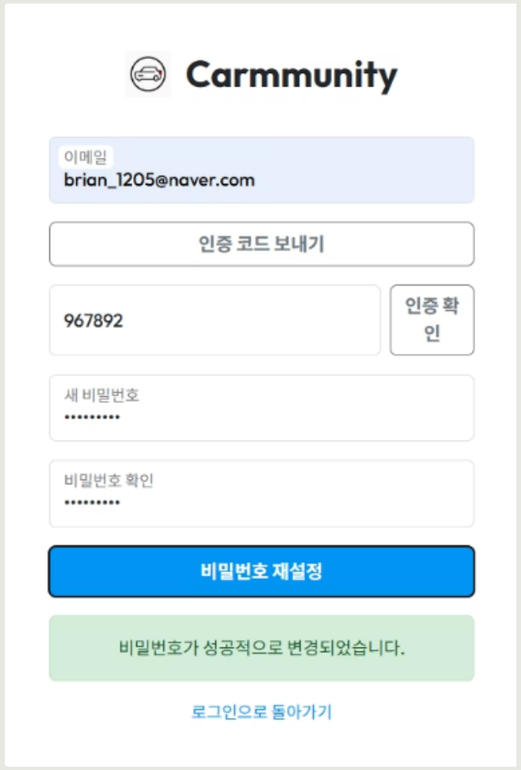
```java
@Service  3 usages  Sungwon Lee *
public class CommentService {

    private final CommentRepository commentRepository;  10 usages
    private final CommentLikeRepository commentLikeRepository;  4 usages
    private final PostRepository postRepository;  2 usages
    private final UserService userService;  7 usages

    public CommentService(CommentRepository commentRepository,  Sungwo
                          CommentLikeRepository commentLikeRepository,
                          PostRepository postRepository,
                          UserService userService) {
        this.commentRepository = commentRepository;
        this.commentLikeRepository = commentLikeRepository;
        this.postRepository = postRepository;
        this.userService = userService;
    }
```

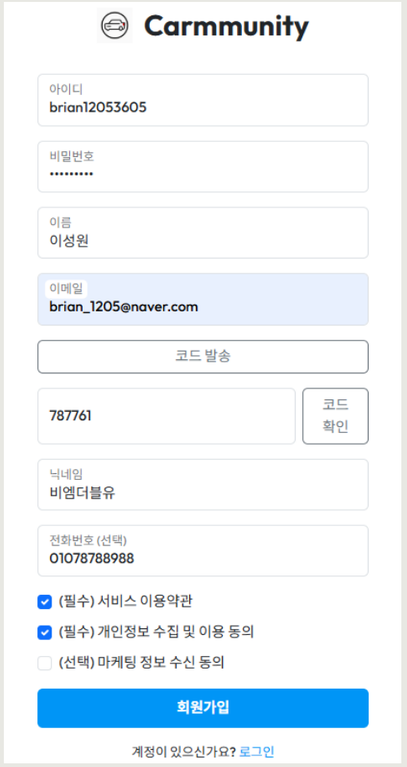
```java
@Service  3 usages  Sungwon Lee *
@RequiredArgsConstructor
public class CommentService {

    private final CommentRepository commentRepository;
    private final CommentLikeRepository commentLikeRepository;
    private final PostRepository postRepository;
    private final UserService userService;
```

주요 기능 - 회원 관리

이메일 인증 기반 회원가입

안전한 회원 인증 및 관리 시스템 구현



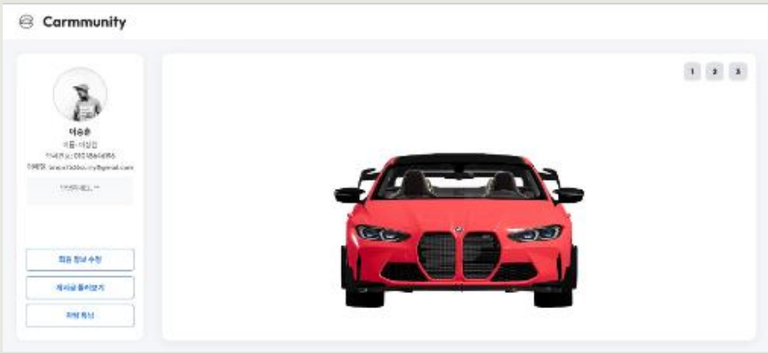


프로필 이미지 선택

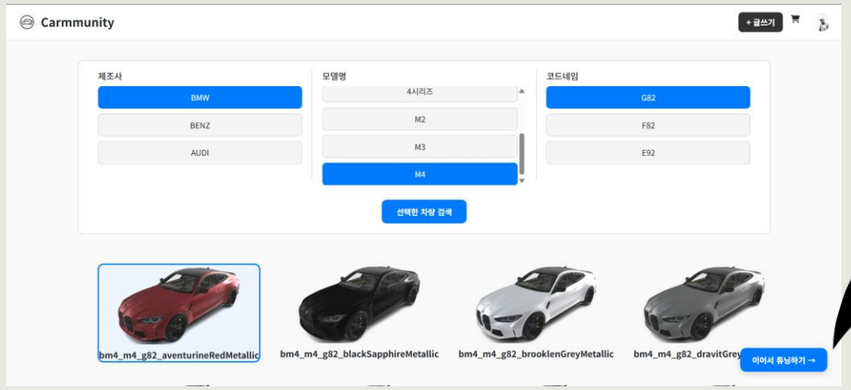
사용자 맞춤형 프로필 설정 가능

튜닝 슬롯 저장 기능

사용자 튜닝 정보 저장 및 불러오기

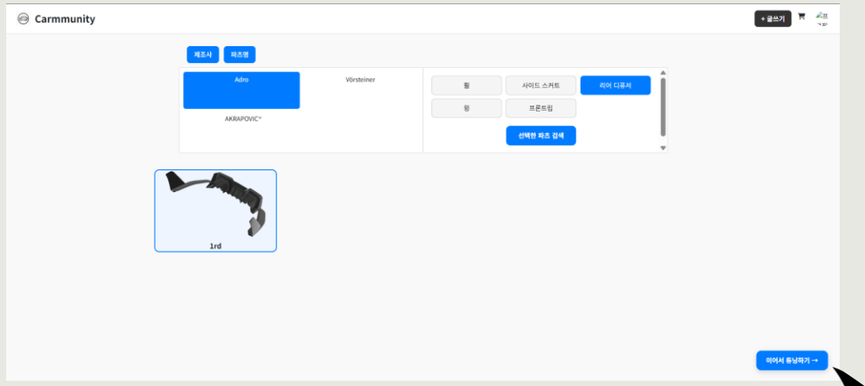


주요 기능 – 차량 튜닝



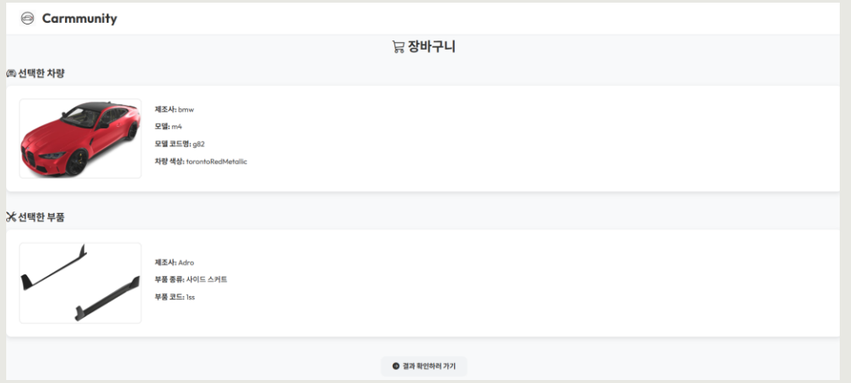
1단계: 차량 선택

다양한 차량 모델 중 선택



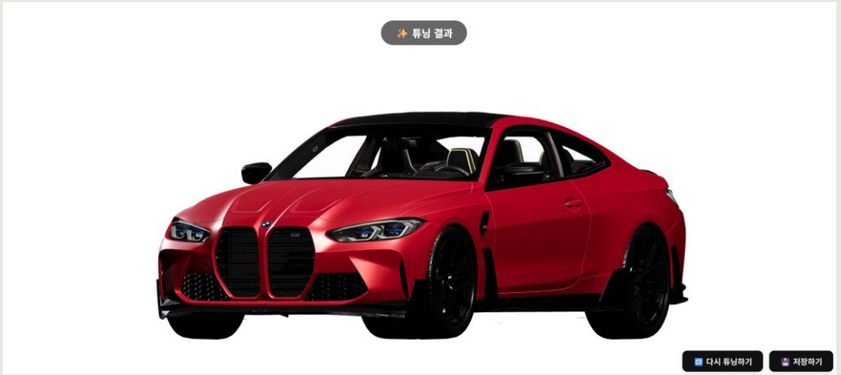
2단계: 부품 선택

호환되는 튜닝 부품 탐색



3단계: 결과 확인

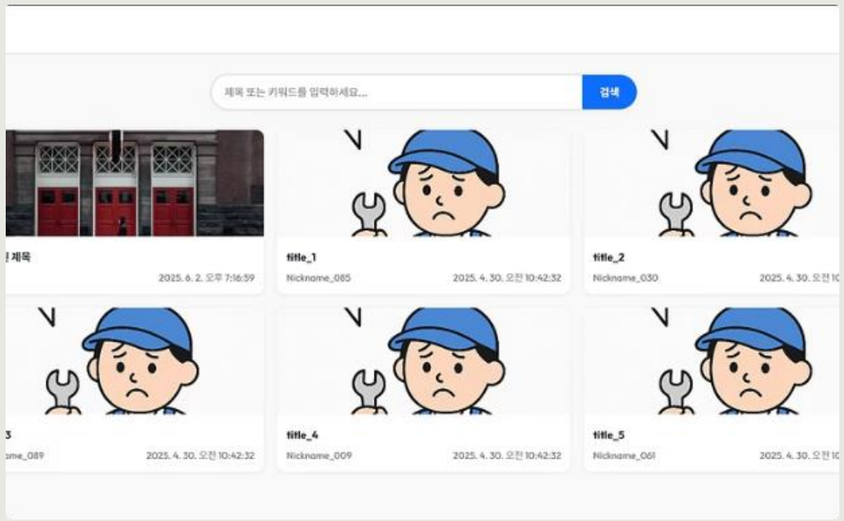
3D 시각화로 튜닝 결과 미리보기



4단계: 장바구니

선택한 부품 목록 확인

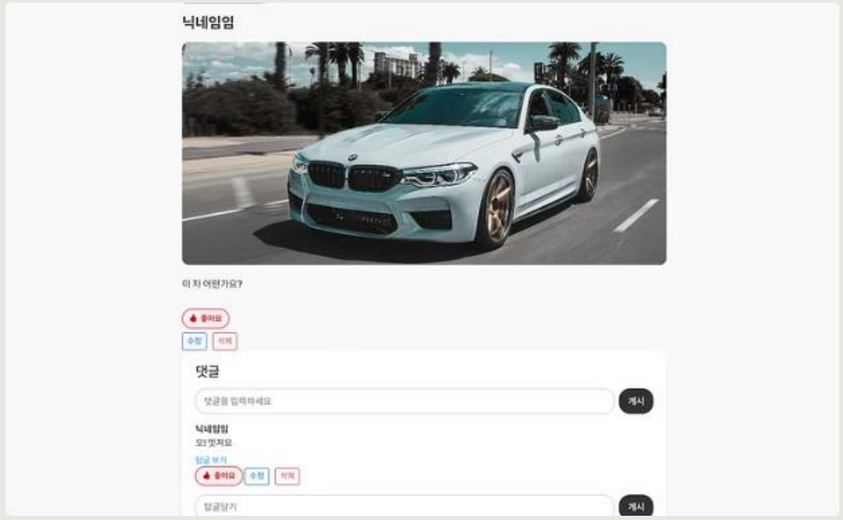
주요 기능 - 게시글 관리



게시글 목록

썸네일 미리보기 기능

무한 스크롤로 편리한 탐색



게시글 상세

댓글 페이징 처리

좋아요 동시성 제어

회고록

역할 분담 문제

명확한 경계 설정의 어려움

테스트 환경 미비

초기 테스트 인프라 부족

초기 공통 로직 부재

중복 코드 발생

부족한 테스트 케이스

예상치 못한 버그 발생

추가 구현 사항

비동기 서버 최적화

성능 향상

소셜 로그인 도입

사용자 편의성 증대

통합 테스트 환경 구축

안정성 확보

개발 규칙 및 공통 로직 정의

효율적인 협업

테스트 전략 및 예외 케이스 대응

견고한 시스템 구축

감사합니다