

MAGE Platform

Team B2



- Name: Peter B
- Repositories:
 - <https://github.com/B2MAGE/mage-backend>
 - <https://github.com/B2MAGE/mage-frontend>
- Artifacts:
 - <https://github.com/orgs/B2MAGE/projects/1> (currently private)
 - <https://github.com/B2MAGE/mage-backend/blob/main/README.md>
 - <https://github.com/B2MAGE/mage-backend/blob/main/CONTRIBUTING.md>
 - <https://github.com/B2MAGE/mage-backend/tree/main/docs>
 - <https://github.com/B2MAGE/mage-frontend/blob/main/README.md>

MAGE Platform

At-a-glance



What I worked on

- Created frontend and backend repository
- Set up GitHub Projects board for SCRUM workflow
- Configured branch protection for main
 - Required pull requests to merge to main
 - Required one code review before merge
- Initialized React + TypeScript + Vite frontend
- Added React Router routing and initial routes
- Built shared frontend layout component
- Implemented header navigation
- Set up backend project structure
- Configured database connectivity
- Configured database migrations
- Created user persistence layer
- Implemented User model and repository
- Implemented Google authentication endpoint (POST /auth/google)
- Wrote onboarding/setup documentation for the team

MAGE Platform

Team B2



Example Code:

```
@RestController
@RequestMapping("/auth")
public class AuthController {

    private final GoogleAuthenticationService googleAuthenticationService;

    public AuthController(GoogleAuthenticationService googleAuthenticationService) {
        this.googleAuthenticationService = googleAuthenticationService;
    }

    @PostMapping("/google")
    ResponseEntity<GoogleAuthenticationResponse> authenticateWithGoogle(
        @Valid @RequestBody GoogleAuthenticationRequest request) {
        GoogleAuthenticationResult result = this.googleAuthenticationService.authenticate(request.getIdToken());
        HttpStatus status = result.created() ? HttpStatus.CREATED : HttpStatus.OK;

        return ResponseEntity.status(status)
            .body(new GoogleAuthenticationResponse(
                result.user().getId(),
                result.user().getEmail(),
                result.user().getDisplayName(),
                result.user().getAuthProvider().name(),
                result.created()));
    }
}
```

AuthController:

- Exposes the POST /auth/google route
- Accepts a validated Google ID token request
- Delegates authentication to the service layer
- Returns the Google-backed user
 - 201 Created for new accounts
 - 200 OK for existing ones

MAGE Platform

Team B2



Example Artifact:

A screenshot of a Jira Sprint Board for 'MAGE Platform Development'. The board is organized into three columns: 'Backlog' (4 items), 'Ready' (19 items), and 'In Progress' (6 items). Each item is a task card with a title, sprint information, and priority. The 'Backlog' column contains tasks like 'Create user login endpoint', 'Implement authentication middleware', and 'Create user profile endpoint'. The 'Ready' column contains tasks like 'Create preset creation endpoint', 'Create preset retrieval endpoint', and 'Create user preset list endpoint'. The 'In Progress' column contains tasks like 'Create user registration endpoint', 'Create tags database table', and 'Create presets database table'. The board also shows a search bar with 'sprint:"Sprint 1"', navigation tabs for 'Sprint Board', 'Backlog by Epic', 'Team Workload', 'By Area', 'By Epic', 'Blocked Items', and 'New view', and a search filter for 'sprint:"Sprint 1"'. Each task card includes a green progress indicator, a title, sprint and epic information, a priority level (P0, P1), and a count of items.

GitHub Project Page:

- Manage the MAGE development workflow
- Track sprint tasks and progress
- Organize work across development stages
- Coordinate team collaboration
- Link issues and pull requests to tasks
- Monitor blockers and dependencies

MAGE Platform

Team B2



Long Project Topic Idea:

Monolith vs Microservices: Architectural Tradeoffs for Small Development Teams

Key Sections:

- Monolithic architecture overview
- Microservices architecture overview
- Tradeoffs (complexity, scaling, DevOps)
- Case studies from startups
- When small teams should or should not use microservices