Email Badge Combiner: A Modern Web Application for OZ Digital Recognition System

OZ Digital Recognition System Overview

October 19, 2025

1 Project Overview

The Email Badge Combiner is a sophisticated web application designed to streamline the process of combining achievement badges and core value recognitions for OZ Digital's internal recognition system. Deployed on Azure Static Web Apps, it leverages modern web technologies to provide a reliable and scalable solution for badge generation.

2 Purpose and Use Case

The application serves two primary purposes:

- Combining Ozpert achievement badges (Bronze, Silver, Gold, Diamond) with OZ Digital's core value badges.
- Creating standardized, professional-looking composite images for use in email communications and recognition systems.

The tool enables users to:

- Select from multiple achievement levels.
- Combine them with core value recognitions.
- Generate consistent, properly sized composite images.
- Download results for immediate use.

3 Technical Implementation

3.1 Architecture

The application is built using:

- **Frontend**: HTML5, CSS3, and vanilla JavaScript.
- **Image Processing**: HTML5 Canvas API.
- **Deployment**: Azure Static Web Apps.
- **CI/CD**: GitHub Actions.
- **Development Tools**: Node.js and npm.

3.2 Key Components

3.2.1 Badge Generation System

- Uses Node.js Canvas API to generate placeholder badges.
- Implements dynamic text wrapping for badge labels.
- Maintains consistent 200x200 pixel dimensions.

3.2.2 Image Processing Engine

- Implements smart image resizing while maintaining aspect ratios.
- Centers images in standardized spaces.
- Provides clean white backgrounds for consistency.

3.2.3 User Interface

- Modern, responsive design.
- Intuitive radio button selection interface.
- Real-time canvas preview.
- Background styling with Teams theme integration.

3.3 Deployment

The application is deployed using Azure Static Web Apps with automated CI/CD through GitHub Actions, including:

- Automatic builds triggered by pushes to the main branch.
- Node.js dependency management.
- Static file serving optimization.
- Custom routing configurations.

4 Features

4.1 Badge Selection

- Ozpert Badges: Bronze, Silver, Gold, Diamond.
- Core Values: Be Positive, Empathize, Evolve Constantly, Focus on Outcome, We Succeed as a Team.

4.2 Image Processing

- Automatic image resizing.
- Aspect ratio preservation.
- Consistent output dimensions.
- Clean white backgrounds.

4.3 User Experience

- Real-time preview.
- One-click download.
- Modern, responsive interface.

- Teams-themed background.

4.4 Error Handling

- Input validation.
- Image loading error management.
- User feedback for actions.

5 How to Replicate

5.1 Prerequisites

- Node.js (latest LTS version).
- npm package manager.
- Azure account.
- GitHub account.

5.2 Setup Steps

5.2.1 Azure Deployment

1. Create a new Static Web App in Azure Portal.
2. Connect to your GitHub repository.
3. Configure build settings:
    - App location: ”/”.
    - Output location: ”/”.
    - Build command: ”npm run build:azure”.

6 Potential Improvements

6.1 Technical Enhancements

- Add image caching for improved performance.
- Implement drag-and-drop interface for custom badges.
- Add batch processing capabilities.
- Include image optimization before download.

6.2 Feature Additions

- Custom badge text input.
- Additional badge combinations (3+ badges).
- Badge templates and presets.

- Share functionality for direct email integration.

6.3 User Experience

- Preview animations for selection.
- Mobile-optimized interface.
- Dark mode support.
- Accessibility improvements.

6.4 Administrative Features

- Badge management interface.
- Usage analytics.
- Custom color schemes.
- Template management.

7 Maintenance and Support

The application requires minimal maintenance, including:

- Regular updates to dependencies.
- Monitoring of Azure Static Web App metrics.
- Periodic review of badge designs and colors.
- Updates to core values or achievement levels as needed.

8 Best Practices Implemented

8.1 Code Organization

- Modular JavaScript functions.
- Clear separation of concerns.
- Consistent naming conventions.
- Comprehensive error handling.

8.2 Performance

- Efficient image processing.
- Optimized canvas operations.
- Minimal dependencies.
- Responsive design practices.

8.3 Security

- Content security policies.
- Secure file handling.
- Azure security best practices.
- Input validation.

8.4 Deployment

- Automated CI/CD pipeline.
- Environment configuration.
- Build optimization.
- Error logging.

9 Conclusion

The Email Badge Combiner is a crucial tool in OZ Digital's recognition system, streamlining the creation of professional badge combinations while maintaining consistency and ease of use. Its modern architecture and deployment on Azure ensure reliability and scalability for future growth. Implementing the suggested improvements will further enhance its functionality and user experience.

For additional details or assistance, please contact OZ Digital's IT department or visit xAI's website.