Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to define the requirements for the Notification Board system. This system will enable users to post and view content such as news, job advertisements, and general announcements. The system will display this content on a carousel interface accessible via web browsers and TV screens, primarily located in office gates and lobbies.

1.2 Scope

The Notification Board system is a cloud-based application designed to:

- Allow Admins and Editors to create and manage content.
- Provide public visibility of content via a carousel interface.
- Support responsiveness for mobile, TV, and large screens.
- Facilitate media uploads and category-based content organization.

The system will enhance communication and information dissemination within organizations, utilizing TV screens (40 inches or larger) and mobile-friendly access.

1.3 Definitions, Acronyms, and Abbreviations

- Admin: A user role with full permissions to manage the system, including user approvals and configurations.
- Editor: A user role with permissions to create and schedule content.
- Carousel: A rotating display of content.

2. System Features

2.1 User Roles and Permissions

1. Admin:
    - Manage user accounts (create, edit, delete).
    - Manage posts (create, edit, delete).
    - Assign user approval requirements. If a user needs some to approve his post or he can directly.
    - Adjust carousel transition settings (default: 10 seconds, adjustable).
2. Editor:
    - Create and manage posts, including uploading images and PDFs.
    - Specify visibility duration for posts.
    - Schedule posts.
3. Public Users:
    - View content via the website or TV screens.

4. Authenticated Users:
    - Comment on posts after signing up with their email.
    - Provide an OTP sent to their email for signup and login.
    - Use their email as a username for commenting.
    - View comments only when logged in.

2.2 Content Management

- Users can create posts with the following components:
    - Text
    - Images (multiple per post)
    - PDFs
- Maximum media size: 20MB.
- Content can be categorized for organizational purposes.
- Media should be readable and downloadable.

2.3 Carousel Display

- Content will be displayed in a carousel interface.
- Default transition time: 10 seconds (modifiable by Admin).
- Users can define the visibility duration of their posts (e.g., 1 day, 1 week).

2.4 Scheduling

- Users can schedule posts for future display.

2.5 Responsiveness

- The system will adapt to various screen sizes, including:
    - Mobile devices
    - TVs (40 inches or larger)
    - Large desktop screens

2.6 Security and Access Control

- Public access to view content.
- If feasible, implement access control for TV screens to restrict unauthorized modifications.

2.7 User Comments

- Users can comment on posts after providing their email address.
- An OTP will be sent to the provided email for verification.
- Email addresses will serve as usernames for commenting.
- Comments will be visible to all logged-in users.
- Users must have an account to view comments.

3. Non-Functional Requirements

3.1 Performance

- The system should handle up to 1,000 concurrent users.
- Carousel transitions must remain smooth under load.

3.2 Scalability

- The system will be cloud-hosted to ensure scalability and accessibility.

3.3 Usability

- Intuitive user interface with clear navigation for Admins and Editors.
- Drag-and-drop or file selector for media uploads.

3.4 Availability

- System uptime should be 99.9%.
- Scheduled maintenance windows will be announced in advance.

3.5 Security

- HTTPS must be enforced for secure communication.
- Access to Admin features will require authentication.
- Implement role-based access control (RBAC).
- OTP-based email verification for account creation and login.

3.6 Data Management

- Store media files securely in the cloud.
- Ensure content and user data integrity with regular backups.

4. Technical Requirements

4.1 Platform

- Cloud-based deployment.
- Support for major browsers (Chrome, Firefox, Edge).

4.2 Media Support

- File types: JPEG, PNG, PDF.
- Max size: 20MB per file.

4.3 TV Display

- Support resolutions commonly used in 40-inch+ TVs.
- Real-time updates or periodic auto-refresh.

4.4 Development Stack

- Backend: Node.js, PHP, .NET, or Spring Boot
- Frontend: React.js, Vue.js, or Angular.
- Database: MySQL or PostgreSQL.

5. Constraints

- No integration with Active Directory.
- Single-language support.
- TV screens will be 40 inches or larger.

6. Assumptions and Dependencies

- All users accessing the system will have reliable internet connections.
- TV screens are already installed and configured.
- Organizational branding guidelines will be provided before development begins.

```
notification-board-backend/
├── cmd/
│   ├── server/
│   │   └── main.go           # Entry point for the application
├── config/
│   └── config.go             # Configuration handling (e.g., environment variables)
├── internal/
│   ├── app/
│   │   ├── app.go            # Application initialization (routes, dependencies)
│   │   ├── middleware.go     # Middleware for authentication, logging, etc.
│   ├── domain/
│   │   ├── models/
│   │   │   ├── post.go       # Post model
│   │   │   ├── user.go       # User model
│   │   │   └── comment.go    # Comment model
│   │   ├── interfaces/
│   │   │   ├── repository.go # Repository interface definitions
│   │   │   └── service.go    # Service interface definitions
│   ├── handlers/
│   │   ├── post_handler.go   # Handlers for post-related APIs
│   │   ├── user_handler.go   # Handlers for user-related APIs
│   │   └── comment_handler.go # Handlers for comment-related APIs
```

```
│   ├── repositories/
│   │   ├── post_repo.go       # Post repository implementation
│   │   ├── user_repo.go       # User repository implementation
│   │   └── comment_repo.go    # Comment repository implementation
│   ├── services/
│   │   ├── post_service.go    # Business logic for posts
│   │   ├── user_service.go    # Business logic for users
│   │   └── comment_service.go # Business logic for comments
│   └── utils/
│       ├── auth.go            # Utility for JWT/OTP handling
│       ├── file_upload.go     # Utility for handling file uploads
│       ├── logger.go          # Logging utility
│       └── response.go        # Helper for standard API responses
├── pkg/
│   ├── db/
│   │   ├── database.go        # PostgreSQL database connection setup
│   │   └── migrations/        # SQL migration files
│   ├── email/
│   │   └── email.go           # Email OTP sending logic
│   ├── scheduler/
│   │   └── scheduler.go       # Scheduling logic for posts
├── docs/
│   ├── api-docs.yaml          # API documentation (Swagger/OpenAPI)
├── go.mod                     # Go module file
├── go.sum                     # Dependency checksum
├── Makefile                   # Task runner for building, testing, and running
```

```
└── README.md                # Project documentation
```