

ATTENDO Documentation

Complete User Guide & System Documentation



Setup Guide

Quick setup instructions



System Architecture

Technical diagrams

Current



Vendor Guide

Daily attendance tracking



Manager Guide

Team management



Admin Guide

System configuration



Notifications

Email & SMS alerts



ATTENDO System Architecture

Complete technical guide to the Vendor Timesheet Management System

Version 2.0

Last Updated: January 2025

MediaTek Hackathon Edition



Database Schema

</> API Reference

📄 Flow Diagrams



Tech Stack



Download Docs

System Overview

ATTENDO is a comprehensive vendor timesheet management system designed for MediaTek's hackathon, providing automated attendance tracking, reconciliation, and notification services.

18+

Core Components

12

Database Tables

35+

API Endpoints

3

User Roles

Key Features

- **Multi-Role System:** Vendor, Manager, and Admin interfaces
- **Real-time Notifications:** Email, SMS, and Microsoft Teams integration
- **Power Automate Integration:** Excel-based workflow automation with intelligent notification flows
- **Attendance Reconciliation:** Automated mismatch detection
- **Comprehensive Reporting:** Monthly reports and analytics
- **Audit Trail:** Complete activity logging
- **Data Import/Export:** CSV/Excel file processing
- **Holiday Management:** Automatic holiday handling
- **AI-Powered Analytics:** Smart predictions and insights
- **Advanced Manager Tools:** Table-based mismatch review interface



Python Flask
Backend Framework



SQLite
Database Engine



Bootstrap 5
Frontend UI



Pandas
Data Processing



APScheduler



Power Automate

Task Scheduling

Excel-Based Notifications



Scikit-Learn

AI/ML Analytics

System Architecture

ATTENDO follows a layered architecture pattern with clear separation of concerns.

System Architecture Layers

Presentation Layer

Vendor
Dashboard

Manager
Dashboard

Manager
Mismatches
Table

Admin Console

API Endpoints

AI Analytics

Documentation

Business Logic Layer

Authentication

Authorization

Notification
Service

Reconciliation
Engine

AI Prediction
Engine

Manager
Analytics

Report
Generator

Excel
Integration

Data Access Layer

SQLAlchemy
ORM

Database
Models

Query Builders

Migration
Scripts

Data Storage Layer

SQLite
Database

Excel Files

Log Files

Configuration
Files

External Integrations

Power Automate
Flow

Excel
Notification

OneDrive Sync

Microsoft Teams

Database Design

The system uses a normalized relational database design with clear entity relationships.

Database Schema (12 Tables)

users

id	<i>INTEGER PRIMARY KEY</i>
username	<i>VARCHAR(80) UNIQUE NOT NULL</i>
email	<i>VARCHAR(120) UNIQUE NOT NULL</i>
password_hash	<i>VARCHAR(255) NOT NULL</i>
role	<i>ENUM(vendor, manager, admin)</i>
is_active	<i>BOOLEAN DEFAULT TRUE</i>
created_at	<i>DATETIME DEFAULT CURRENT_TIMESTAMP</i>
last_login	<i>DATETIME</i>

vendors

id	<i>INTEGER PRIMARY KEY</i>
user_id	<i>INTEGER FK → users.id</i>
vendor_id	<i>VARCHAR(50) UNIQUE NOT NULL</i>
full_name	<i>VARCHAR(100) NOT NULL</i>
department	<i>VARCHAR(100) NOT NULL</i>
company	<i>VARCHAR(100) NOT NULL</i>
band	<i>VARCHAR(10) NOT NULL</i>
location	<i>VARCHAR(50) NOT NULL</i>
manager_id	<i>VARCHAR(50) FK → managers.manager_id</i>

managers

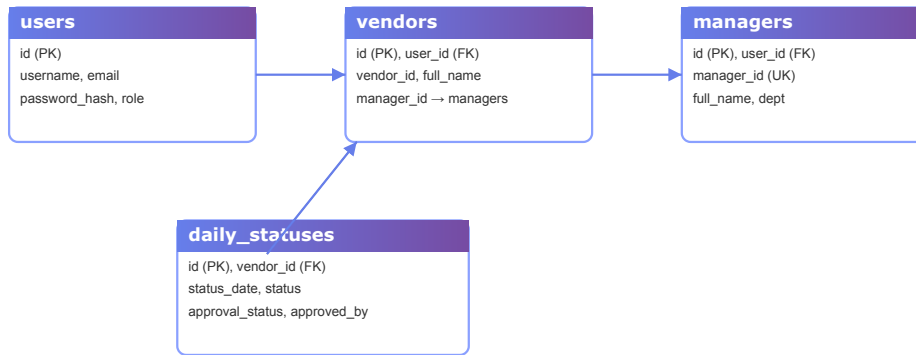
id	<i>INTEGER PRIMARY KEY</i>
manager_id	<i>VARCHAR(50) UNIQUE NOT NULL</i>
user_id	<i>INTEGER FK → users.id</i>
full_name	<i>VARCHAR(100) NOT NULL</i>
department	<i>VARCHAR(100) NOT NULL</i>
team_name	<i>VARCHAR(100)</i>
email	<i>VARCHAR(120)</i>
phone	<i>VARCHAR(20)</i>

daily_statuses

id	<i>INTEGER PRIMARY KEY</i>
vendor_id	<i>INTEGER FK → vendors.id</i>
status_date	<i>DATE NOT NULL</i>
status	<i>ENUM(in_office_full, in_office_half, wfh_full, wfh_half, leave_full, leave_half, absent)</i>
location	<i>VARCHAR(100)</i>
comments	<i>TEXT</i>
approval_status	<i>ENUM(pending, approved, rejected)</i>
approved_by	<i>INTEGER FK → users.id</i>
manager_comments	<i>TEXT</i>
INDEX	<i>idx_vendor_date (vendor_id, status_date)</i>

i Additional Tables: **swipe_records** (attendance machine data), **holidays** (holiday configuration), **mismatch_records** (reconciliation data), **notification_logs** (notification history), **audit_logs** (system audit trail), **system_configurations** (system settings), **leave_records** & **wfh_records** (imported data), **email_notification_logs** (email tracking)

Entity Relationship Diagram (ERD)



Entity Relationships

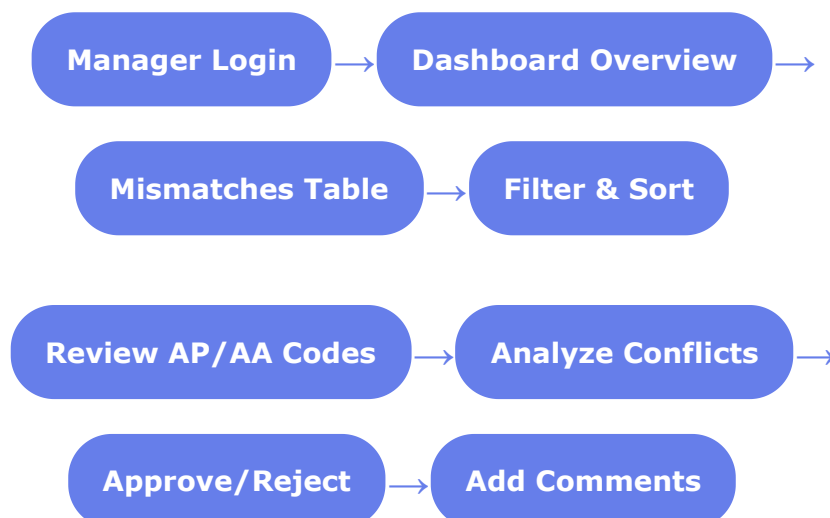
- **Users** → **Vendors/Managers** (1:1 relationship)
- **Managers** → **Vendors** (1:Many relationship)
- **Vendors** → **Daily Statuses** (1:Many relationship)
- **Vendors** → **Swipe Records** (1:Many relationship)
- **Vendors** → **Mismatch Records** (1:Many relationship)
- **Users** → **Audit Logs** (1:Many relationship)

📊 Manager Mismatches Table Interface

Advanced table-based interface for managers to efficiently review and resolve team attendance mismatches with comprehensive filtering and action capabilities.

Key Features & User Flow

Manager Mismatch Resolution Flow



Notify Vendor

Update Records

Generate Reports

Table Interface Features

UI/UX

- **AP/AA Code Legend:** Clear explanations of swipe status codes with visual indicators
- **Advanced Filtering:** Filter by status (pending/approved/rejected), team member, and conflict priority
- **Interactive Tooltips:** Hover tooltips on AP/AA badges explaining "Attendance Present" vs "Attendance Absent"
- **Color-Coded Priority:** High (red), Medium (yellow), Low (blue) conflict classification
- **Direct Actions:** Inline approve/reject buttons with comment prompts
- **Team Member Avatars:** Visual identification with status-based color coding
- **Responsive Design:** Mobile-friendly table with touch-optimized interactions

Smart Conflict Detection

LOGIC

```
# Conflict Priority Classification Algorithm
if (claimed_status in ['wfh_full', 'wfh_half', 'leave_full', 'leave_half'] and swipe_status == 'AP'):
    priority = 'HIGH' # Claimed remote but physically present
elif (claimed_status in ['in_office_full', 'in_office_half'] and swipe_status == 'AA'):
    priority = 'MEDIUM' # Claimed office but not detected
else:
    priority = 'LOW' # Approval or record issues
```

- **High Priority:** Employee claimed WFH/Leave but swipe shows AP (physically present)
- **Medium Priority:** Employee claimed office attendance but swipe shows AA (absent)
- **Low Priority:** Approval workflow or record keeping issues

Navigation & Integration

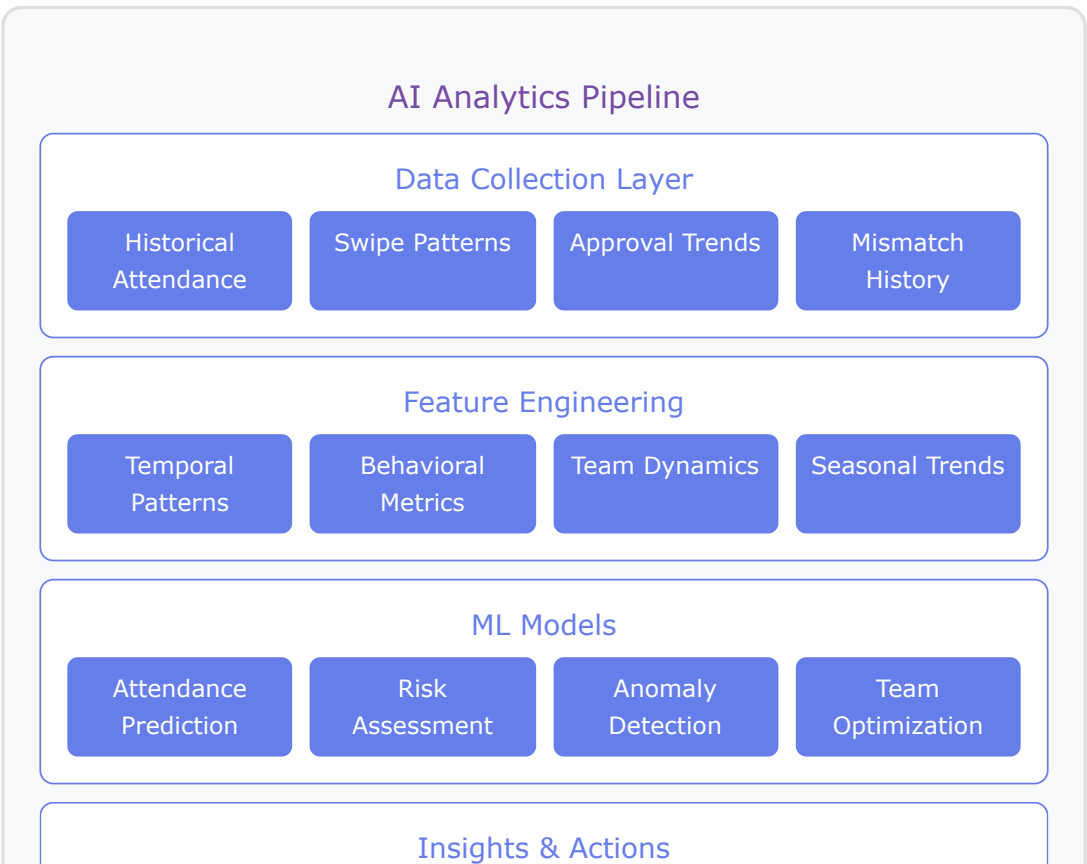
- **Dashboard Integration:** Direct "Mismatches Table" button in manager quick actions
- **Per-Vendor Links:** Table view filtered by specific team member from main dashboard
- **Breadcrumb Navigation:** Clear path between dashboard, card view, and table view
- **Export Capabilities:** Generate filtered reports in Excel format
- **Real-time Updates:** Immediate reflection of approval actions across all views

💡 **Manager Experience Enhancement:** The table interface provides the same functionality as the admin reconciliation page but filtered specifically for the manager's team, ensuring role-appropriate data access and streamlined workflow.

🧠 AI-Powered Analytics & Predictions

Advanced artificial intelligence features providing predictive analytics, behavioral insights, and automated recommendations for attendance management.

AI Engine Architecture



AI Features & Capabilities

Predictive Analytics

PREDICTION

- **Attendance Prediction:** Forecast likely WFH/office days for each team member
- **Mismatch Prediction:** Identify vendors at high risk of attendance conflicts
- **Team Capacity Planning:** Predict office occupancy and team availability
- **Seasonal Pattern Analysis:** Identify trends based on holidays, weather, projects

```
# AI Prediction Example def
predict_attendance_risk(vendor_id, date_range): features
= extract_behavioral_features(vendor_id) risk_score =
ml_model.predict_proba(features) return { 'risk_level':
'HIGH' if risk_score > 0.7 else 'MEDIUM' if risk_score >
0.4 else 'LOW', 'confidence': risk_score,
'recommendations': generate_recommendations(features) }
```

Behavioral Analysis

ANALYSIS

- **Pattern Recognition:** Identify individual and team attendance patterns
- **Anomaly Detection:** Flag unusual attendance behaviors for review
- **Compliance Scoring:** Track adherence to company attendance policies
- **Performance Correlation:** Analyze attendance vs productivity patterns

Manager Insights Dashboard

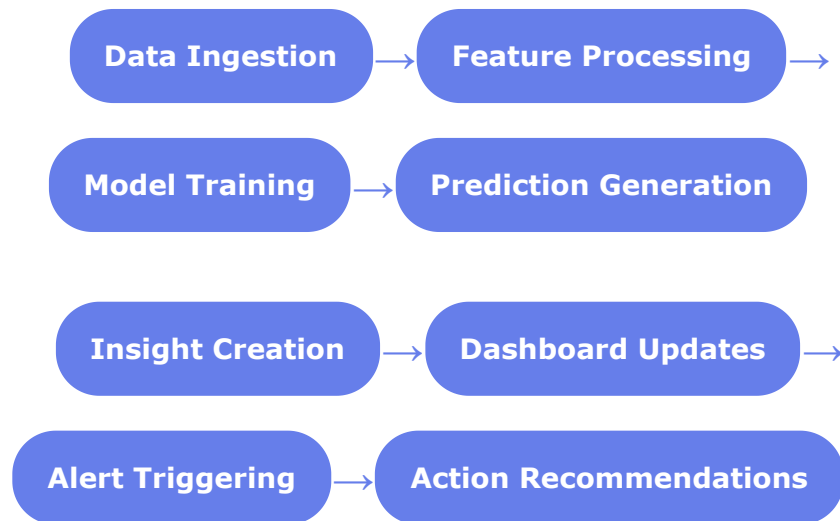
INSIGHTS

- **Team Performance Metrics:** AI-powered team efficiency and attendance analytics
- **Predictive Alerts:** Early warning system for potential attendance issues
- **Optimization Recommendations:** AI suggestions for improving team attendance policies

- **Resource Planning:** Predicted office space and resource requirements

AI Model Integration

AI Analytics Pipeline Flow



AI Technology Stack: Built using Python's scikit-learn for machine learning models, pandas for data processing, and integrated with the main Flask application for real-time predictions and insights delivery.

System Components

The system is built using modular components for maintainability and scalability.

Core Application Files

Backend Components

- **app.py** - Main Flask application
- **models.py** - Database models & schemas
- **utils.py** - Utility functions
- **import_routes.py** - Data import endpoints
- **notifications.py** - Notification system
- **swagger_ui.py** - API documentation

Integration Components

- **power_automate_api.py** - Power Automate integration
- **excel_table_formatter.py** - Excel table handling
- **daily_excel_updater.py** - Real-time Excel updates
- **notification_service.py** - Enhanced notifications
- **realtime_notification_sync.py** - Real-time sync

Key Component Features

Authentication & Authorization

SECURITY

- Flask-Login for session management
- Role-based access control (Vendor, Manager, Admin)
- Password hashing with Werkzeug
- Session timeout and security headers

Notification System

MESSAGING

- Multi-channel notifications (Email, SMS, Teams)
- Configurable notification templates
- Real-time Excel-based triggers
- Notification scheduling and queueing

Data Processing Engine

PROCESSING

- Pandas-based Excel/CSV processing
- Automated mismatch detection algorithms
- Monthly report generation
- Data import/export functionality

Power Automate Integration

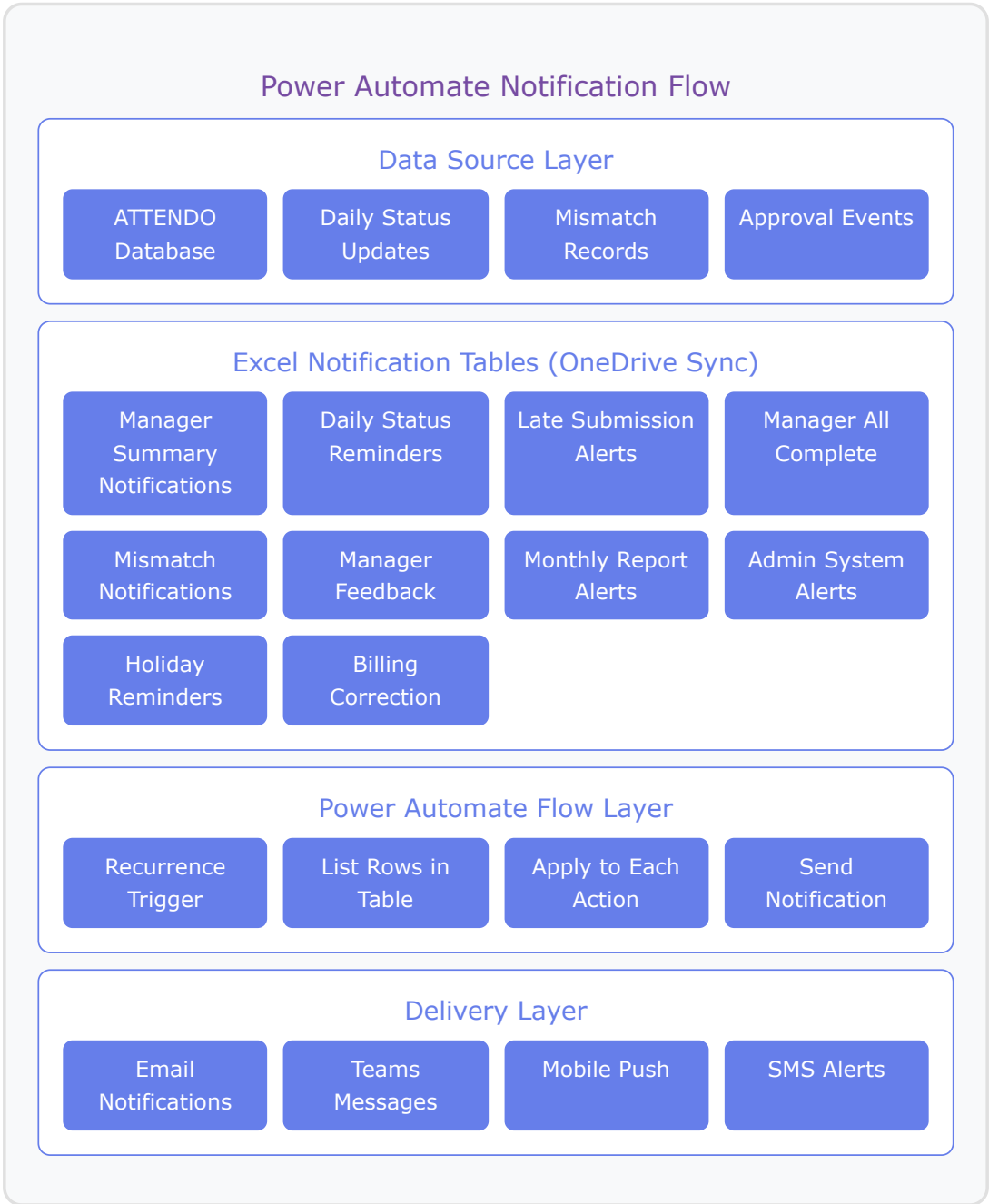
INTEGRATION

- Excel table-based data exchange
- Real-time webhook notifications
- Configurable automation workflows
- Error handling and retry logic

■ Power Automate Notification System

Advanced notification system using Microsoft Power Automate with Excel-based workflows for intelligent, recurrent notifications to relevant users.

Notification Architecture Overview



Excel-Based Notification Tables

Notification Table Structure EXCEL

The system maintains 10+ Excel notification tables synchronized with OneDrive for Power Automate consumption:

- **02_manager_summary_notifications:** Daily summary of team status for managers
- **01_daily_status_reminders:** Automated reminders for pending status submissions

- **09_late_submission_alerts:** Alerts for overdue attendance submissions
- **03_manager_all_complete_notifications:** Notifications when all team members have submitted
- **04_mismatch_notifications:** Alerts for attendance reconciliation conflicts
- **05_manager_feedback_notifications:** Manager approval/rejection notifications
- **06_monthly_report_notifications:** Monthly attendance report distribution
- **07_admin_system_alerts:** System administration notifications
- **08_holiday_reminder_notifications:** Holiday and special day reminders
- **10_billing_correction_notifications:** Billing reconciliation alerts

Power Automate Workflow Design

Notification Flow Structure



Integration Benefits

FEATURES

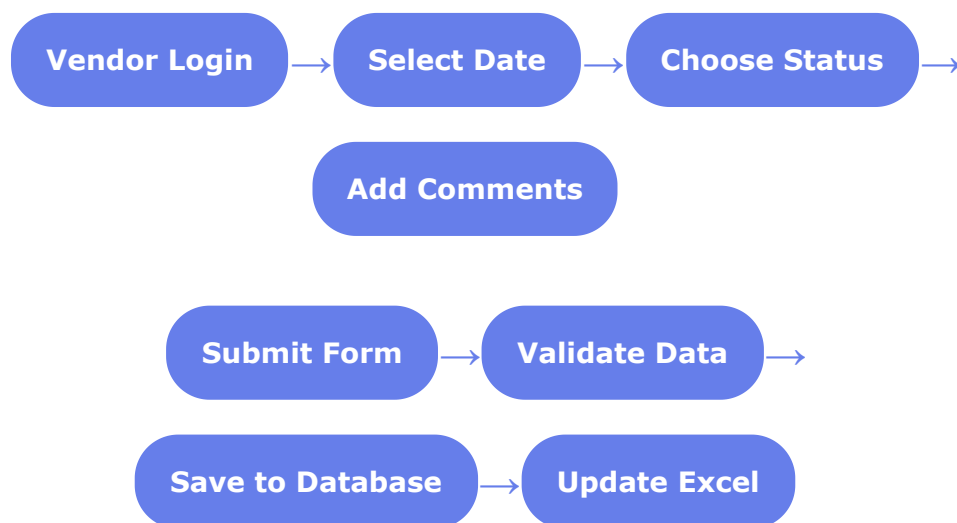
- **Serverless Operation:** No infrastructure management required for notifications
- **Real-time Sync:** OneDrive ensures immediate availability of notification data
- **Scalable Processing:** Power Automate handles variable notification volumes
- **Intelligent Filtering:** Excel formulas enable smart recipient targeting
- **Multi-channel Delivery:** Single flow supports email, Teams, SMS delivery
- **Audit Trail:** Complete notification history in Excel format
- **Easy Configuration:** Non-technical users can modify notification templates
- **Cost Effective:** Uses existing Microsoft 365 infrastructure

i Implementation Approach: The system keeps notification data synchronized in OneDrive Excel files, enabling Power Automate flows to pick up changes and deliver targeted notifications to relevant users through recurrent polling of Excel table data.

Data Flow Diagrams

Understanding how data flows through the system for different processes.

Vendor Status Submission Flow



Trigger Notifications →

Update Power Automate →

Create Audit Log

Manager Approval Flow

Manager Dashboard →

View Pending →

Review Details →

Approve/Reject

Update Status →

Add Comments →

Notify Vendor →

Update Excel

Reconciliation Process Flow

Import Swipe Data →

Parse Excel/CSV →

Match Vendors

Compare Statuses →

Detect Mismatches →

Create Mismatch Records

Notify Managers

Generate Reports

Notification System Flow

Status Change Event

Update Excel Tables

Trigger Power Automate

Check Notification Rules

Generate Messages

Send Notifications

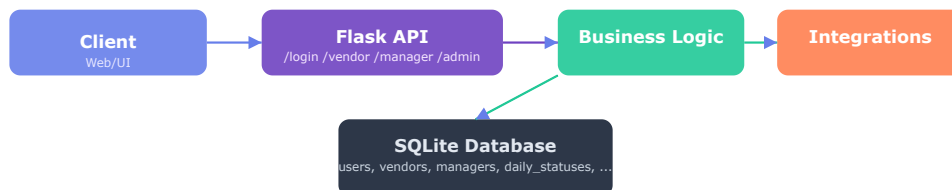
Log Delivery Status

Handle Failures

🔌 API Design

RESTful API endpoints for all system operations with comprehensive Swagger documentation.

Endpoint Map



Authentication Endpoints

POST /login

POST

User authentication with username/password validation

GET /logout

GET

User session termination and cleanup

Vendor Endpoints

POST /vendor/submit-status

POST

Submit daily attendance status with validation and real-time Excel updates

PUT /vendor/update-status/{id}

PUT

Update existing attendance status with audit logging

Manager Endpoints

POST /manager/approve-status/{id}

POST

Approve or reject vendor status submissions

POST /manager/approve-status/bulk

POST

Bulk approval/rejection of multiple status entries

GET /manager/mismatches/table

GET

Advanced table view for team attendance mismatches with filtering by status, vendor, and conflict priority

POST /manager/mismatch/{id}/approve

POST

Approve or reject individual mismatch explanations with manager comments

Admin Endpoints

POST /admin/add-holiday

POST

Add new holiday dates to the system calendar

POST /admin/import-swipe-data

POST

Import attendance machine data from Excel/CSV files

AI Analytics Endpoints

GET /manager/ai-insights

GET

AI-powered team insights dashboard with predictive analytics and recommendations

GET /api/ai/predict-attendance/{vendor_id}

GET

Generate attendance predictions and risk assessments for individual team members

GET /api/ai/team-analytics

GET

Comprehensive team performance analytics with behavioral insights and trends

Power Automate Integration Endpoints

GET /power-automate/vendors


GET

Retrieve vendor data in Power Automate compatible format

POST /power-automate/webhook

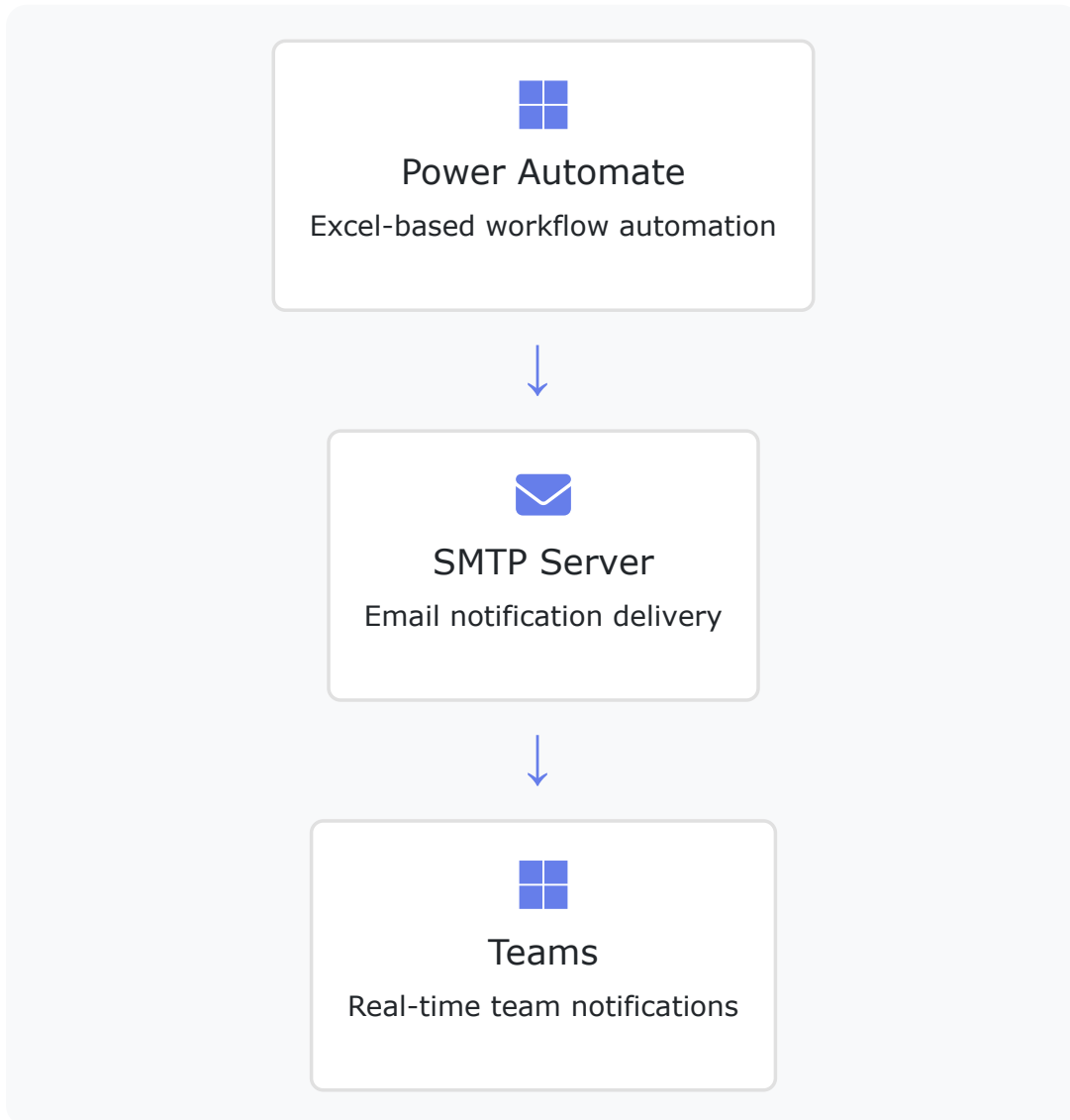
POST

Receive webhook notifications from Power Automate workflows

 **API Documentation:** Complete API documentation is available at </docs> endpoint with interactive Swagger UI for testing all endpoints. New manager mismatches table and AI analytics endpoints are included with comprehensive parameter documentation.

External Integrations

The system integrates with multiple external services for comprehensive functionality.



Integration Details

Microsoft Power Automate

WORKFLOW

- **Excel Tables:** Data structured as named tables for easy reference
- **Real-time Updates:** Immediate Excel file updates on status changes
- **Webhook Integration:** Bi-directional communication with Power Automate
- **Error Handling:** Robust retry logic and error notifications

```
# Excel table update on vendor status change def  
handle_vendor_status_submission(vendor_id):
```

```
update_notification_table(df, 'daily_reminders')
trigger_power_automate_webhook(vendor_id, 'SUBMITTED')
```

Email & SMS Notifications

MESSAGING

- **SMTP Integration:** Configurable email server settings
- **Template System:** HTML/text email templates
- **SMS Gateway:** Optional SMS API integration
- **Delivery Tracking:** Comprehensive notification logging

Microsoft Teams

COLLABORATION

- **Webhook Integration:** Direct channel notifications
- **Rich Messages:** Formatted cards with action buttons
- **Team Channels:** Manager-specific team notifications
- **Real-time Alerts:** Immediate mismatch and urgency alerts

❗ Configuration Required: Set environment variables for SMTP_SERVER, SMTP_USER, SMTP_PASSWORD, SMS_API_URL, and Teams webhook URLs for full integration functionality.

🛡️ Security Architecture

Multi-layered security approach ensuring data protection and access control.

Authentication & Authorization

- **Session-based Authentication:** Flask-Login for secure session management
- **Password Security:** Werkzeug password hashing with salt
- **Role-based Access Control:** Three-tier permission system (Vendor/Manager/Admin)
- **Session Timeout:** Automatic session expiration for security

Data Protection

- **Input Validation:** Server-side validation for all form inputs
- **SQL Injection Protection:** SQLAlchemy ORM with parameterized queries
- **CSRF Protection:** Flask-WTF CSRF token validation
- **XSS Prevention:** Template escaping and content sanitization

Audit & Monitoring

- **Comprehensive Audit Trail:** All user actions logged with timestamps
- **IP Address Tracking:** Request origin logging for security monitoring
- **User Agent Logging:** Browser and device information capture
- **Failed Login Monitoring:** Brute force attack detection

⚠ **Security Considerations:** In production, implement HTTPS, database encryption, rate limiting, and consider multi-factor authentication for enhanced security.

⚙ Scalability & Performance

System designed for growth and high-performance operations.

Current Architecture Strengths

- **Modular Design:** Loosely coupled components for easy scaling
- **Database Indexing:** Optimized queries with strategic indexes
- **Efficient Data Processing:** Pandas for high-performance data operations
- **Background Tasks:** APScheduler for non-blocking operations

Scaling Strategies

Database Scaling

- Migration to PostgreSQL/MySQL
- Read replicas for reporting
- Database connection pooling
- Query optimization

Application Scaling

- Load balancer deployment
- Horizontal scaling with containers
- Redis for session storage
- CDN for static assets

Performance Optimizations

- **Database Indexes:** Strategic indexing on frequently queried columns
- **Query Optimization:** Efficient SQLAlchemy queries with minimal N+1 problems
- **Caching Strategy:** Template caching and query result caching
- **Async Processing:** Background job processing for heavy operations

💡 **Performance Monitoring:** Implement application performance monitoring (APM) tools and database query analysis for production deployments.

📊 Monitoring & Logging

Comprehensive monitoring and logging system for operational excellence.

Application Monitoring

- **Real-time Sync Monitoring:** Excel file sync status and health checks
- **Notification Delivery Tracking:** Success/failure rates for all channels
- **System Health Metrics:** Database connections, response times
- **Error Tracking:** Exception logging and alerting

Logging Architecture

- **Structured Logging:** JSON-formatted logs with consistent fields
- **Log Levels:** DEBUG, INFO, WARNING, ERROR, CRITICAL classification
- **Audit Logs:** Database-stored audit trail for compliance
- **Application Logs:** File-based logging with rotation

```
# Example logging configuration logging.basicConfig(
level=logging.INFO, format='%(asctime)s - %(name)s - %(levelname)s
- %(message)s', handlers=[ logging.FileHandler('attendo.log'),
logging.StreamHandler() ] )
```

📌 **Monitoring Tools:** Consider implementing Prometheus/Grafana for metrics, ELK stack for log analysis, and Sentry for error tracking in production.

🚀 Deployment Architecture

Flexible deployment options for different environments and scales.

Development Environment

```
# Quick start for development python app.py # System runs on
```

Production Deployment Options

Traditional Server Deployment

SERVER

- Gunicorn + Nginx for production WSGI serving
- PostgreSQL/MySQL for production database
- Redis for session storage and caching
- systemd for service management

Container Deployment

DOCKER

- Docker containerization for consistent deployments
- Docker Compose for multi-service orchestration
- Kubernetes for enterprise-scale deployments
- Volume mounting for persistent data storage

Cloud Deployment

CLOUD

- Azure App Service for easy cloud deployment
- AWS Elastic Beanstalk for managed infrastructure
- Google Cloud Run for serverless containers
- Cloud databases for managed data storage

Environment Configuration

```
# Required environment variables FLASK_ENV=production
SECRET_KEY=your-secret-key
DATABASE_URL=postgresql://user:pass@host/db
SMTP_SERVER=smtp.gmail.com SMTP_USER=your-email@domain.com
SMTP_PASSWORD=your-app-password
```

Deployment Checklist

- ☒ Environment variables configured
- ☒ Database migrations applied
- ☒ HTTPS certificates installed
- ☒ Backup strategy implemented
- ☒ Monitoring tools configured
- ☒ Security hardening applied
- ☒ Load testing completed
- ☒ Disaster recovery plan in place

⚠ Production Readiness: Ensure proper security configuration, database optimization, and monitoring setup before production deployment.

© 2025 ATTENDO - System Architecture Documentation
Developed for MediaTek Hackathon | Technical Reference Guide