Hotel Booking System Documentation

Table of Contents

Project Overview

Description

A modern hotel reservation system built with Flask backend and responsive HTML/CSS/JavaScript frontend. The system uses Google Sheets as a database for storing room information, pricing data, and bookings.

Key Features

- Dynamic pricing based on dates and occupancy
- Real-time availability checking
- Multi-step booking process with progress tracking
- Responsive design for mobile and desktop
- Google Sheets integration for data management
- Automated booking confirmation system

Technology Stack

- **Backend**: Python Flask, Google Sheets API
- **Frontend**: HTML5, CSS3, JavaScript (ES6+)
- **Database**: Google Sheets

- **Authentication**: OAuth2 Service Account
- **Styling**: Custom CSS with animations and gradients

System Architecture

High-Level Architecture

```
Frontend (HTML/CSS/JS) → Flask Backend → Google Sheets API → Google Sheets Database
```

Components

1. **Frontend Layer**: User interface with booking forms and room displays
2. **API Layer**: Flask routes handling business logic
3. **Data Layer**: Google Sheets serving as database
4. **Authentication Layer**: OAuth2 service account for secure API access

---

Installation & Setup

Prerequisites

- Python 3.7+
- Google Cloud Platform account
- Google Sheets API enabled
- Service account credentials

Installation Steps

1. **Clone/Download the project files**
2. **Install Python dependencies**

```bash
pip install flask flask-cors gspread oauth2client
```

3. **Set up Google Sheets API**
   - Create a Google Cloud Project
   - Enable Google Sheets API
   - Create a Service Account
   - Download the credentials JSON file

- Place it in your project directory

4. **Update configuration**
   - Replace the credentials file path in `app.py`
   - Update the `SPREADSHEET_ID` in `app.py`

5. **Create project structure**

```
project/
├── app.py
├── templates/
│   └── index.html
├── static/
│   ├── css/
│   │   └── style.css
│   ├── js/
│   │   └── main.js
│   └── images/
└── credentials.json
```

Database Setup (Google Sheets)

Sheet Structure

Your Google Sheets document must contain exactly 3 worksheets:

1. Rooms Sheet

Column	Type	Description
Room_Type	Text	Unique room type identifier
Max_Occupancy	Number	Maximum guests allowed
Base_Price	Number	Base price for display
Description	Text	Room description
Image_URL	Text	Image filename

Example Data:

```
Room_Type | Max_Occupancy | Base_Price | Description | Image_URL
Deluxe    | 4             | 150        | Spacious room with city view | deluxe.jpg
Suite     | 6             | 250        | Luxury suite with balcony | suite.jpg
Standard  | 2             | 100        | Comfortable standard room | standard.jpg
```

2. Pricing Sheet (Critical for Date Ranges)

Column	Type	Description
Date	Date (YYYY-MM-DD)	Specific date
Room_Type	Text	Must match Room_Type from Rooms sheet
Single_Rate	Number	Price for 1 adult
Double_Rate	Number	Price for 2 adults
Extra_Person	Number	Additional cost per extra adult
Extra_Child	Number	Additional cost per child
With_Breakfast	Number	Breakfast supplement cost
Available_Rooms	Number	Rooms available for this date

Example Data:

```
Date       | Room_Type | Single_Rate | Double_Rate | Extra_Person | Extra_Child | With_Breakfast | Available_Rooms
2024-01-01 | Deluxe    | 120         | 150         | 30           | 15          | 25             | 10
2024-01-01 | Suite     | 200         | 250         | 40           | 20          | 25             | 5
2024-01-02 | Deluxe    | 120         | 150         | 30           | 15          | 25             | 8
2024-01-02 | Suite     | 200         | 250         | 40           | 20          | 25             | 3
```

3. Bookings Sheet

Column	Type	Description
Booking_ID	Number	Auto-generated unique ID
Guest_Name	Text	Guest full name
Email	Text	Guest email address
Phone	Text	Guest phone number
Check_In	Date	Check-in date
Check_Out	Date	Check-out date
Room_Type	Text	Booked room type
Adults	Number	Number of adults
Children	Number	Number of children
Breakfast	Boolean	Breakfast included
Total_Amount	Number	Total booking cost
Status	Text	Booking status

Database Population Strategy

Method 1: Manual Entry (Small Hotels)

- Enter dates manually for 6-12 months ahead

- Update monthly with new dates

Method 2: Formula-Based (Recommended)

Use Google Sheets formulas to generate date ranges:

1. **Generate Date Column:**

```
=SEQUENCE(365, 1, DATE(2024,1,1))
```

2. **Create Room Type Repetitions:** For each date, create entries for all room types using array formulas.
3. **Set Dynamic Pricing:** Use formulas to adjust prices based on:

- Day of week (weekends higher)
- Seasons (peak/off-peak)
- Special events

Method 3: Script-Based (Advanced)

Create a Google Apps Script to:

- Generate date ranges automatically
- Update availability based on bookings
- Implement dynamic pricing rules

Backend API Documentation

Base URL

`http://localhost:5000`

Endpoints

GET /api/rooms

Description: Retrieve all room types and their details **Response**:

```
json
```

```
[
  {
    "Room_Type": "Deluxe",
    "Max_Occupancy": 4,
    "Base_Price": 150,
    "Description": "Spacious room with city view",
    "Image_URL": "deluxe.jpg"
  }
]
```


POST /api/pricing

Description: Get pricing data for date range **Request Body**:


```
json

{
  "check_in": "2024-01-01",
  "check_out": "2024-01-03"
}
```


Response: Array of pricing data filtered by date range

POST /api/availability

Description: Check room availability for date range **Request Body**:


```
json

{
  "check_in": "2024-01-01",
  "check_out": "2024-01-03"
}
```


Response:


```
json

{
  "Deluxe": 5,
  "Suite": 2
}
```


POST /api/calculate-price

Description: Calculate total price for booking **Request Body**:

```json
{
  "check_in": "2024-01-01",
  "check_out": "2024-01-03",
  "room_type": "Deluxe",
  "adults": 2,
  "children": 1,
  "breakfast": true
}
```

Response:

```json
{
  "total_price": 425.50,
  "nights": 2,
  "price_per_night": 212.75
}
```

POST /api/book

Description: Create a new booking **Request Body**:

```json
{
  "guest_name": "John Doe",
  "email": "john@example.com",
  "phone": "+1234567890",
  "check_in": "2024-01-01",
  "check_out": "2024-01-03",
  "room_type": "Deluxe",
  "adults": 2,
  "children": 1,
  "breakfast": true,
  "total_amount": 425.50
}
```

Response:

```json
```

```
{
  "success": true,
  "booking_id": 123,
  "message": "Booking confirmed successfully!"
}
```

Frontend Components

Navigation Header

- Fixed position navigation
- Smooth scrolling to sections
- Responsive design

Hero Section

- Full-screen background with gradient overlay
- Call-to-action button
- Animated entrance effects

Search Section

- Date picker with validation
- Guest selection (adults/children)
- Form validation and error handling

Rooms Grid

- Dynamic room cards with pricing
- Availability indicators
- Responsive layout

Booking Modal

Three-step process:

1. **Details & Extras**: Booking summary and add-ons
2. **Guest Information**: Personal details form
3. **Payment & Confirmation**: Price summary and terms

Key JavaScript Functions

Modal Navigation

- `nextStep(stepNumber)`: Move to next step with validation
- `prevStep(stepNumber)`: Go back to previous step
- `updateProgressIndicators(activeStep)`: Update visual progress

Data Management

- `loadRooms()`: Fetch room data from API
- `handleSearch(e)`: Process search form and update results
- `updatePriceSummary()`: Calculate and display pricing

Booking Process

- `openBookingModal(roomType, basePrice)`: Initialize booking flow
- `handleBooking(e)`: Submit booking to backend
- `validateCurrentStep()`: Ensure step completion before proceeding

Booking Flow

User Journey

1. **Landing**: User arrives at homepage
2. **Search**: Select dates, guests, search for rooms
3. **Browse**: View available rooms with pricing
4. **Select**: Choose room and click "Book Now"
5. **Details**: Review booking, select extras
6. **Information**: Enter guest details
7. **Payment**: Review total, agree to terms, confirm
8. **Confirmation**: Receive booking ID and details

Data Flow

1. Frontend collects search criteria
2. Backend queries Google Sheets for pricing/availability
3. Frontend displays filtered results
4. User selects room, modal opens with booking details
5. User proceeds through 3-step process
6. Backend creates booking record in Google Sheets
7. System returns confirmation

File Structure

```
hotel-booking-system/
├── app.py                    # Flask backend application
├── credentials.json          # Google API credentials (not in repo)
├── templates/
│   └── index.html            # Main HTML template
├── static/
│   ├── css/
│   │   └── style.css         # Main stylesheet
│   ├── js/
│   │   └── main.js           # Frontend JavaScript
│   └── images/               # Room images directory
│       ├── deluxe.jpg
│       ├── suite.jpg
│       └── standard.jpg
└── README.md                 # Project documentation
```

Configuration

Environment Variables

Create a `.env` file for production:

```
FLASK_ENV=production
SECRET_KEY=your-secret-key
GOOGLE_SHEETS_ID=your-spreadsheet-id
```

Google Sheets Configuration

1. **Spreadsheet ID**: Found in the Google Sheets URL
2. **Worksheet Names**: Must be exactly "Rooms", "Pricing", "Bookings"
3. **Permissions**: Service account must have edit access to the spreadsheet

Frontend Configuration

Update these variables in `main.js`:

```javascript
const API_BASE_URL = '/api';  // Change for production
const IMAGE_BASE_URL = '/static/images/';
```

Troubleshooting

Common Issues

Modal Steps Not Working

Problem: Continue button doesn't advance steps **Solution**: Ensure `nextStep()` and `prevStep()` functions are properly defined and called

Date Range Not Working

Problem: No pricing data returned for selected dates **Solutions**:

- Verify date format is YYYY-MM-DD in Google Sheets
- Ensure continuous date entries (no gaps)
- Check that room types match exactly between Rooms and Pricing sheets

Authentication Errors

Problem: Google Sheets API returns 401/403 errors **Solutions**:

- Verify service account credentials file path
- Check that service account has edit permissions on the spreadsheet
- Ensure Google Sheets API is enabled in Google Cloud Console

Price Calculation Errors

Problem: Incorrect pricing or missing price data **Solutions**:

- Verify all required columns exist in Pricing sheet
- Check date filtering logic in backend
- Ensure numeric fields are properly formatted

Debug Mode

Enable Flask debug mode for development:

```python
if __name__ == '__main__':
    app.run(debug=True)
```

Maintenance & Updates

Regular Tasks

Daily

- Monitor booking confirmations
- Check for API errors in logs

Weekly

- Update pricing data for upcoming dates
- Review availability numbers
- Backup booking data

Monthly

- Extend pricing data for future months
- Archive old booking records
- Update room information if needed

Database Maintenance

Adding New Room Types

1. Add entry to Rooms sheet
2. Add corresponding pricing entries for all future dates
3. Update frontend if new amenities/features

Updating Pricing

1. Modify existing entries in Pricing sheet
2. Add seasonal pricing adjustments
3. Set special event pricing

Managing Availability

- Reduce Available_Rooms when bookings are made
- Increase when cancellations occur
- Block dates for maintenance

Performance Optimization

Backend

- Implement caching for room data
- Optimize Google Sheets queries
- Add database connection pooling

Frontend

- Compress and minify CSS/JavaScript
- Optimize images
- Implement lazy loading for room cards

Security Considerations

Data Protection

- Never commit credentials to version control
- Use environment variables for sensitive data
- Implement input sanitization

API Security

- Add rate limiting
- Implement request validation
- Use HTTPS in production

Deployment

Production Checklist

- [ ] Update Flask configuration for production
- [ ] Set up proper error logging
- [ ] Configure reverse proxy (nginx/Apache)
- [ ] Set up SSL certificates
- [ ] Update Google Sheets permissions
- [ ] Test all booking flows
- [ ] Set up monitoring and alerts

Environment Setup

```python
# Production app.py configuration
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=False)
```

API Error Codes

Code	Description	Solution
400	Bad Request	Check request format and required fields
401	Unauthorized	Verify Google API credentials
403	Forbidden	Check Google Sheets permissions
404	Not Found	Verify endpoint URLs
500	Internal Server Error	Check server logs and database connection

---

Support & Contact

Development Team

- **Backend Developer**: Flask API and Google Sheets integration
- **Frontend Developer**: UI/UX and JavaScript functionality
- **Database Administrator**: Google Sheets structure and maintenance

Documentation Updates

This documentation should be updated whenever:

- New features are added
- API endpoints change
- Database structure modifications
- Bug fixes are implemented

Last Updated: August 28, 2025 Version: 1.0