TSF Travel Management System - 30-Day Sprint Plan

Overview

Based on the new module breakdown (Travel Approval + Travel Claim + Employee Reimbursement), here's an aggressive but achievable development plan to complete the Travel module within 30 days, with expense module as stretch goal.

Week 1: Foundation & Core Structure

Days 1-2: Project Restructuring

Deliverable: Clean, modular codebase structure

Day 1 Tasks:

1. **Create new folder structure**

```
apps/
├── authentication/
├── master_data/
├── travel/
├── expenses/
└── notifications/
```

2. **Split models.py** (6-8 hours)
   - Move User, Role, Permission models → `apps/authentication/models.py`
   - Move geography models → `apps/master_data/models/geography.py`
   - Move travel models → `apps/travel/models/application.py`
   - Create `__init__.py` files with proper imports

Day 2 Tasks:

1. **Reorganize views and serializers** (6-8 hours)
   - Split by domain (auth, master_data, travel)
   - Fix all imports
   - Test basic functionality
2. **Enhanced User model implementation**

```
python
```

```
# Add to User model
reporting_manager = models.ForeignKey('self', ...)
base_location = models.ForeignKey('master_data.Location', ...)
```

Days 3-4: Enhanced Permission System

Deliverable: Working multi-role dashboard system

Day 3 Tasks:

1. **Implement enhanced Role/Permission models** (4 hours)
   - Add dashboard_access field to Role
   - Add is_primary field to UserRole
   - Add permission categories
2. **Create role management utilities** (4 hours)

```
python

# User model methods
def get_primary_role(self)
def can_access_dashboard(self, dashboard_type)
def get_approval_hierarchy(self)
```

Day 4 Tasks:

1. **Update authentication views** (4 hours)
   - Enhanced login response with role information
   - Role switching API endpoint
   - Dashboard routing logic
2. **Permission mixins and decorators** (4 hours)
   - HasPermissionMixin for views
   - Custom permission validators

Days 5-7: Master Data Models

Deliverable: All master data models and APIs

Day 5 Tasks:

1. **Travel-specific master models** (6 hours)

```
python
```

```
class ApprovalMatrix(models.Model):
    # Amount-wise approval rules
class GuestHouseMaster(models.Model):
    # TSF guest houses
class ARCHotelMaster(models.Model):
    # Approved hotels
class LocationSPOC(models.Model):
    # Vehicle booking coordinators
```

Day 6 Tasks:

1. **DA/Incidental calculation models** (4 hours)

```
python

class DAIncidentalMaster(models.Model):
    # Grade-wise DA rates by city category
```

2. **Master data APIs and serializers** (4 hours)
   - CRUD operations for all master models
   - Proper validation rules

Day 7 Tasks:

1. **Testing and bug fixes** (6 hours)
   - Test all master data APIs
   - Fix import issues
   - Ensure proper permissions
2. **Database migration and sample data** (2 hours)
   - Create sample grades, locations, travel modes
   - Test user hierarchy setup

Week 2: Travel Application Core

Days 8-10: Enhanced Travel Models

Deliverable: Complete travel application system

Day 8 Tasks:

1. **Enhanced TravelApplication model** (4 hours)

```
python
```

```
# Add required TSF fields
internal_order = models.CharField(max_length=50)
general_ledger = models.CharField(max_length=50)
sanction_number = models.CharField(max_length=50)
advance_amount = models.DecimalField(...)
```

2. **Accommodation and Vehicle booking models** (4 hours)

```python
class AccommodationBooking(models.Model):
    # Guest house priority logic
class VehicleBooking(models.Model):
    # SPOC-based vehicle requests
```

Day 9 Tasks:

1. **Business logic validators** (6 hours)

```python
# apps/travel/business_logic/validators.py
def validate_advance_booking(departure_date, travel_mode)
def validate_flight_fare(amount, grade)
def validate_own_car_distance(distance)
```

2. **Travel application serializers** (2 hours)
   - Enhanced validation
   - Nested booking serializers

Day 10 Tasks:

1. **Travel application APIs** (6 hours)
   - Create/Read/Update travel requests
   - File upload handling
   - Status management
2. **Basic testing** (2 hours)
   - Test travel request creation
   - Validate business rules

Days 11-14: Approval Workflow Engine

Deliverable: Dynamic approval system

Day 11 Tasks:

1. **Approval workflow models** (4 hours)

```python
class TravelApprovalFlow(models.Model):
    # Dynamic approval chains
```

2. **Approval engine business logic** (4 hours)

```python
class ApprovalEngine:
    def generate_approval_flow(self)
    def create_approval_records(self)
```

Day 12 Tasks:

1. **Manager approval APIs** (6 hours)
   - List pending approvals
   - Approve/reject actions
   - Approval history tracking
2. **CEO approval for flights >₹10k** (2 hours)
   - Special approval trigger
   - Email notifications

Day 13 Tasks:

1. **Travel desk approval system** (6 hours)
   - Booking coordination
   - File upload for tickets/confirmations
   - Status updates
2. **SPOC vehicle coordination** (2 hours)
   - Location-based SPOC assignment
   - Vehicle request forwarding

Day 14 Tasks:

1. **Integration testing** (6 hours)
   - End-to-end approval flow testing
   - Different user scenarios

- Permission validation

2. **Bug fixes and optimizations** (2 hours)

Week 3: Booking & Document Management

Days 15-17: Booking Process Implementation

Deliverable: Complete booking workflow

Day 15 Tasks:

1. **Accommodation priority logic** (4 hours)
   - Guest House → ARC Hotel → Alternative Hotel
   - Availability checking
   - Automatic fallback
2. **Document management system** (4 hours)

```python
class TravelDocument(models.Model):
    # Booking confirmations, bills, receipts
```

Day 16 Tasks:

1. **Vehicle booking coordination** (6 hours)
   - Local vs inter-unit vehicles
   - SPOC notification system
   - Duty slip generation
2. **Email notification system** (2 hours)
   - Booking confirmations
   - Approval notifications

Day 17 Tasks:

1. **File upload and management** (4 hours)
   - Secure file handling
   - Document validation
   - Storage organization
2. **Booking APIs completion** (4 hours)
   - All booking types (flight, train, hotel, vehicle)
   - Status tracking

Days 18-21: Travel Claim (TA/DA) Module

Deliverable: Basic expense claim system

Day 18 Tasks:

1. **Travel claim models** (4 hours)

```python
class TravelClaim(models.Model):
    # Link to travel application
class ConveyanceClaim(models.Model):
    # Individual expense items
class DACalculation(models.Model):
    # Auto-calculated DA/incidentals
```

2. **DA/Incidental calculation engine** (4 hours)

```python
def calculate_da_incidentals(travel_application)
def validate_conveyance_claims(claims)
```

Day 19 Tasks:

1. **Self-declaration functionality** (4 hours)
   - Expense entry forms
   - Receipt upload
   - Duplicate claim prevention
2. **Claim validation logic** (4 hours)
   - Policy compliance checking
   - Amount validation against entitlements

Day 20 Tasks:

1. **Account verification workflow** (6 hours)
   - Finance team review process
   - Approval/rejection with comments
   - Settlement tracking
2. **Claim APIs** (2 hours)
   - CRUD operations for claims
   - Status management

Day 21 Tasks:

1. **Integration with travel applications** (4 hours)
   - Link claims to approved travel
   - Auto-populate travel details
   - Settlement deadline tracking
2. **Testing and validation** (4 hours)
   - End-to-end claim process
   - Business rule validation

Week 4: Completion & Polish

Days 22-24: Cancellation & Modification

Deliverable: Complete travel lifecycle management

Day 22 Tasks:

1. **Travel modification system** (6 hours)

```python
class TravelModification(models.Model):
    # Change requests with re-approval
class TravelCancellation(models.Model):
    # Cancellation with refund tracking
```

2. **Re-approval workflow** (2 hours)
   - Modified requests need fresh approvals
   - Partial cancellation handling

Day 23 Tasks:

1. **Cancellation APIs** (4 hours)
   - Cancel with reason
   - Refund calculation
   - Booking cancellation coordination
2. **Advanced features** (4 hours)
   - Travel history
   - Reporting dashboards
   - Search and filtering

Day 24 Tasks:

1. **Employee reimbursement module** (6 hours)

```python
class ReimbursementRequest(models.Model):
    # Non-travel expense claims
```

2. **Integration preparation** (2 hours)
   - SAP integration endpoints
   - Data export functionality

Days 25-27: Testing & Bug Fixes

Deliverable: Production-ready travel module

Day 25 Tasks:

1. **Comprehensive testing** (8 hours)
   - All user scenarios
   - Permission boundaries
   - Error handling

Day 26 Tasks:

1. **Performance optimization** (4 hours)
   - Database query optimization
   - API response times
   - File upload efficiency
2. **Security audit** (4 hours)
   - Permission checking
   - Input validation
   - File upload security

Day 27 Tasks:

1. **Bug fixes and refinements** (8 hours)
   - Address all issues found in testing
   - UI/API consistency
   - Error message improvements

Days 28-30: Documentation & Deployment Prep

Deliverable: Deployable system with documentation

Day 28 Tasks:

1. **API documentation** (4 hours)
   - Endpoint documentation
   - Request/response examples
   - Authentication guide
2. **Admin panel customization** (4 hours)
   - Master data management
   - User role assignment
   - System configuration

Day 29 Tasks:

1. **Deployment configuration** (4 hours)
   - Production settings
   - Environment variables
   - Database migration scripts
2. **Final testing** (4 hours)
   - Production-like environment testing
   - Load testing basics

Day 30 Tasks:

1. **Final delivery preparation** (8 hours)
   - Code cleanup
   - Final documentation
   - Demo preparation

Success Metrics

Minimum Viable Product (Travel Module):

- ✅ User authentication with role switching
- ✅ Travel request creation with validation
- ✅ Multi-level approval workflow
- ✅ Booking process (all modes)

- ✅ Document management
- ✅ Basic travel claims

Stretch Goals (If time permits):

- ✅ Employee reimbursement module
- ✅ Advanced reporting
- ✅ SAP integration preparation

Risk Mitigation

High-Risk Areas:

1. **Approval workflow complexity** - Start early, test frequently
2. **File upload/management** - Use proven libraries, implement gradually
3. **Business rule validation** - Create comprehensive test cases

Contingency Plans:

1. **If behind schedule** - Prioritize core travel request/approval flow
2. **If bugs emerge** - Daily bug triage, fix critical issues immediately
3. **If scope creep** - Document additional requirements for post-delivery phase

Development Tips for 30-Day Sprint:

1. **Daily Progress Tracking**: End each day with working, tested code
2. **Incremental Testing**: Test each component as you build it
3. **Code Reuse**: Use Django's built-in features wherever possible
4. **Focus on Core**: Don't get distracted by nice-to-have features
5. **Documentation**: Document as you code, not at the end