# Subscription Management Dashboard
# Software Engineer Intern - Take Home Assessment

aajil SA

September 2025

⚠ Important: Your Thought Process Matters Most

**What we truly care about is YOUR thought process and technical decision-making.**

- We know anyone can send this to ChatGPT/Claude and get a working solution
- We want to see **how YOU think** through the problem
- Document your architectural decisions - why did you choose this approach?
- Show us your iterations, debugging process, and learning journey
- Include comments explaining your logic and trade-offs
- Walk us through your technical choices step-by-step
- If you use AI tools, be transparent and show how you validated, modified, or improved their suggestions

**We value authentic problem-solving and understanding over copy-pasted perfection!**

## 1 Assessment Overview

Build a **Subscription Management Dashboard** to track subscription services with renewal reminders and cost analysis. This reflects a real-world problem many individuals and businesses face in managing recurring expenses.

## 2 Business Context

Subscription services have become ubiquitous - from Netflix to software tools. Users need a centralized way to:

- Track all their subscriptions in one place
- Get alerts before renewals to avoid unexpected charges
- Analyze spending patterns and identify savings opportunities
- Make informed decisions about which services to keep or cancel

Your solution will demonstrate your ability to build practical, user-focused applications with both backend logic and frontend presentation.

Show Your Journey

**Document your development process:**

- What was your initial approach? Why?
- What challenges did you encounter?
- How did you debug and solve problems?
- What design patterns did you choose and why?
- What would you do differently with more time?
- What trade-offs did you make between features and time?

## 3 Core User Stories

1. As a user, I can add a subscription (e.g., Netflix 15.99 Riyal/month) and see when it renews next
2. As a user, I can view my total monthly subscription costs at a glance
3. As a user, I can see which subscriptions are renewing soon (next 7 days)
4. As a user, I can compare monthly vs yearly billing costs for savings opportunities
5. As a user, I can cancel subscriptions I no longer need

## 4 Technical Requirements

### 4.1 Backend (Django + Django REST Framework)

**Models Required:**

- `Subscription` model with fields:
  - name, cost, billing_cycle (monthly/yearly)
  - start_date, renewal_date, is_active
  - category (optional)
- Auto-calculate `renewal_date` based on `billing_cycle` and `start_date`
- **Document why you chose this data model structure**

**API Endpoints:**

- `GET /api/subscriptions/` - List all active subscriptions
- `POST /api/subscriptions/` - Add new subscription
- `PUT/PATCH /api/subscriptions/{id}/` - Update subscription
- `DELETE /api/subscriptions/{id}/` - Cancel subscription

- `GET /api/subscriptions/stats/` - Cost analytics
- **Explain your API design choices and RESTful patterns**

**Validation Rules:**

- Renewal date cannot be in the past
- Cost must be positive
- Billing cycle must be 'monthly' or 'yearly'
- **Show how you handle edge cases and errors**

### 4.2 Frontend (Your Choice: React, Vue, or Django Templates)

Decision Point

Document WHY you chose your frontend framework:

- What are the trade-offs?
- How does it fit the requirements?
- What alternatives did you consider?

**Dashboard Features:**

- Display total monthly spend and projected yearly cost
- Show upcoming renewals in calendar or list view
- Cost breakdown visualization (use any charting library)
- CRUD operations for subscriptions

**Alert System:**

- Highlight subscriptions renewing within 7 days
- Visual indicators for high-cost subscriptions

**Savings Calculator:**

- Show potential savings when switching billing cycles
- Example: "Switch Netflix to yearly and save 120 Riyal/year"

## 5 Implementation Guidelines

Best Practices We're Looking For

- Clean, readable code with meaningful names
- Proper separation of concerns
- Error handling and user feedback
- Basic responsive design
- Comments explaining logic

- Use Django annotations/aggregations for calculations
- Include 3-5 sample subscriptions for demonstration
- Focus on core functionality first, then enhance
- **Document each major decision in your DECISIONS.md file**

## 6 Deliverables

### 6.1 Required Files

1. **Source Code** (GitHub repository preferred)
2. **README.md** - Setup instructions and overview
3. **DECISIONS.md** - Your technical journey:
    - Architecture decisions and rationale
    - Challenges faced and solutions
    - AI tools used and how you validated their output
    - Trade-offs made and why
    - What you learned during the process
    - Future improvements you'd make
4. **requirements.txt** or package.json

### 6.2 Setup Instructions Must Include

- Installation steps
- Database setup
- How to run the application
- Any credentials or configuration needed

## 7 Evaluation Criteria

How We'll Evaluate Your Work

- **Thought Process (35%):** Documentation of decisions and reasoning
- **Functionality (25%):** Does it work as specified?
- **Code Quality (20%):** Clean, maintainable, well-structured
- **Problem Solving (10%):** How you approached challenges
- **User Experience (10%):** Intuitive and pleasant to use

Final Reminder

**Be Authentic:** We're looking for developers who can think critically and communicate their process. If you used AI tools, that's perfectly fine - just show us how you validated, adapted, and understood their suggestions. What matters is YOUR understanding and ability to make informed technical decisions.
Remember: A working solution with great documentation beats a perfect solution with no explanation.

**Good luck! We're excited to see your approach to solving this real-world problem.**