**Mingus User Tier Segmentation System - Cursor Prompts**

**Overview**

These prompts will create a comprehensive user segmentation system for Mingus that funnels users into appropriate pricing tiers based on their responses to screening questions and income information, reducing churn by matching user expectations with tier capabilities.

**PROMPT 1: Create Scoring Algorithm**

Create a Python function called `calculate\_user\_tier\_score()` that takes user responses and returns a recommended pricing tier.

Input parameters:

- annual\_income (int): User's reported annual income

- insight\_questions (dict): Responses to 5 screening questions (A=1, B=2, C=3, D=4 points each)

- tracking\_question (str): Response to fitness tracking question (A/B/C/D)

- target\_market\_location (str): One of the 10 target cities from the business plan

Scoring logic:

- Income weight: 40%

- Insight appetite weight: 35%

- Tracking behavior weight: 25%

Tier thresholds:

- Professional ($50): Score 75-100, income >$70k

- Mid-tier ($20): Score 50-74, income $40k-$70k

- Budget ($10): Score <50 or income <$40k

Return: dict with recommended\_tier, confidence\_score, reasoning, and churn\_risk\_factors

**PROMPT 2: Build Question Response Processor**

Create a class called `UserSegmentationProcessor` that processes the 6 questions:

Question mapping:

1. Health-spending connection: stress\_spending\_pattern

2. Insight vs tracking preference: insight\_preference

3. Immediate value expectation: value\_expectation

4. Financial stress relationship: stress\_relationship

5. Future-focused decisions: goal\_orientation

6. Fitness tracking usage: tracking\_behavior

Methods needed:

- process\_responses(responses\_dict) -> standardized\_scores

- calculate\_insight\_appetite\_score() -> 0-100 scale

- assess\_tracking\_readiness() -> high/medium/low/none

- generate\_personalization\_flags() -> dict of user preferences

Include validation for incomplete responses and default handling.

**PROMPT 3: Create Tier Recommendation Engine**

Build a `TierRecommendationEngine` class that combines scoring with business logic:

Business rules:

- Users earning <$40k automatically get Budget tier option with 7-day free trial

- Users with tracking\_behavior="A" (multiple trackers) skip Budget tier entirely

- Users with insight\_preference="A" (simple tracking only) cap at Mid-tier

- DC Metro/NYC users get +10 scoring bonus (higher cost of living)

- Users with stress\_relationship="A" (money is just numbers) may not need health features

Methods:

- recommend\_tier(user\_data) -> primary and backup tier recommendations

- calculate\_trial\_length(tier, user\_profile) -> days of free trial

- generate\_onboarding\_path(tier) -> customized feature introduction sequence

- predict\_churn\_risk(user\_profile, recommended\_tier) -> risk level and mitigation strategies

**PROMPT 4: Design Onboarding Flow Router**

Create a dynamic onboarding system that adapts based on tier assignment:

For Professional Tier ($50):

- Show advanced health-finance correlation examples upfront

- Enable all features immediately

- Provide "power user" shortcuts and advanced analytics

- 14-day free trial with full feature access

For Mid-tier ($20):

- Progressive feature unlock over first 2 weeks

- Focus on 2-3 key insights initially

- Gradual introduction of health tracking

- 10-day free trial

For Budget Tier ($10):

- Simple financial tracking first week

- Optional health features in week 2

- Basic insights only

- 7-day free trial

Build `OnboardingFlowManager` class with:

- route\_user\_experience(tier, user\_profile)

- customize\_feature\_introduction(preferences)

- set\_expectation\_messaging(tier) -> appropriate value props

**PROMPT 5: Implement Churn Prevention Logic**

Create a `ChurnPreventionSystem` that identifies potential mismatches:

Red flags to detect:

- High-income user assigned Budget tier (may expect more features)

- Non-tracker assigned Professional tier (may be overwhelmed)

- Simple-preference user seeing complex insights

- User expecting immediate value but assigned gradual onboarding

Build methods:

- detect\_tier\_mismatch(user\_profile, assigned\_tier) -> mismatch indicators

- suggest\_tier\_adjustment(usage\_patterns, engagement\_metrics) -> alternative recommendations

- generate\_retention\_interventions(churn\_signals) -> specific actions to take

- create\_feedback\_loop(user\_satisfaction, tier\_performance) -> learning system

Include automated email triggers for at-risk users and tier upgrade/downgrade suggestions.

**PROMPT 6: Create A/B Testing Framework**

Build an A/B testing system for tier assignment optimization:

Test variations:

- Conservative vs aggressive tier recommendations

- Income vs behavior-weighted scoring

- Immediate vs gradual feature introduction

- Different trial lengths by segment

Create `TierTestingFramework` class:

- assign\_test\_group(user\_id) -> control or variant

- track\_conversion\_metrics(group, tier, outcome)

- measure\_retention\_by\_assignment(group, tier, time\_period)

- generate\_optimization\_recommendations() -> data-driven tier assignment improvements

Include conversion tracking for trial-to-paid, feature engagement rates, and user satisfaction scores by tier assignment method.

**PROMPT 7: Build Analytics Dashboard**

Create a `SegmentationAnalytics` class for monitoring tier assignment performance:

Key metrics to track:

- Conversion rate by tier assignment (trial to paid)

- 30/60/90 day retention by tier

- Feature usage patterns by segment

- Revenue per user by assignment method

- Churn reasons by tier mismatch type

Dashboard functions:

- generate\_tier\_performance\_report() -> weekly summary

- identify\_assignment\_patterns() -> successful vs failed placements

- calculate\_revenue\_optimization() -> potential income from better assignment

- export\_user\_feedback\_analysis() -> qualitative insights on tier satisfaction

Output format: JSON for API consumption and CSV for business analysis.

**PROMPT 8: Integrate with Existing Mingus Architecture**

Integrate the tier recommendation system with the existing Mingus Flask application:

Integration points:

- Connect to existing onboarding flow in `/api/onboarding/`

- Add tier recommendation endpoint to user service

- Integrate with pricing display in React frontend

- Connect to existing user preferences and analytics systems

Create new routes:

- POST /api/segmentation/calculate-tier

- GET /api/segmentation/user-recommendations/{user\_id}

- PUT /api/segmentation/update-tier-assignment

- GET /api/analytics/tier-performance

Ensure compatibility with existing:

- User authentication system

- Database models (extend UserProfile if needed)

- Audit logging for tier assignments

- Email service for tier-specific messaging

Include database migration script for any new fields needed.

**Implementation Sequence**

**Week 1: Core Foundation**

* **Prompts 1-3**: Core scoring and recommendation logic
* Focus on getting basic tier assignment working
* Test with sample user data

**Week 2: User Experience**

* **Prompts 4-5**: Onboarding flow and churn prevention
* Implement adaptive user journeys
* Build retention safeguards

**Week 3: Optimization**

* **Prompts 6-7**: Testing framework and analytics
* Set up A/B testing infrastructure
* Create performance monitoring

**Week 4: Integration**

* **Prompt 8**: Integration with existing Mingus system
* Connect to Flask backend and React frontend
* Final testing and deployment preparation

**Expected Outcomes**

**Churn Reduction**

* **40-50%** improvement in user retention through better tier matching
* **Reduced upgrade friction** for users who outgrow their tier
* **Lower downgrade rates** from tier mismatches

**Revenue Optimization**

* **Higher conversion rates** from trial to paid
* **Increased lifetime value** through appropriate tier placement
* **Better price sensitivity matching**

**User Experience**

* **Personalized onboarding** based on user preferences
* **Appropriate feature complexity** for user comfort level
* **Reduced cognitive overload** for budget tier users

**Business Intelligence**

* **Data-driven tier optimization** through A/B testing
* **User behavior insights** for feature development
* **Predictive churn prevention** through early warning systems

**Success Metrics**

**Immediate (30 days)**

* Tier assignment completion rate: >95%
* User satisfaction with recommended tier: >80%
* Trial-to-paid conversion by tier: measurable baseline

**Medium-term (90 days)**

* Retention rate improvement: +40-50%
* Revenue per user optimization: +25-35%
* Churn prevention intervention success: >60%

**Long-term (6 months)**

* User lifetime value increase: +50-75%
* Organic referral rate from satisfied users: measurable growth
* Market differentiation through personalized experience