🚀 Quenito & Quenita Implementation Roadmap

12-Week Journey to $2,000/Week

Start Date: August 2025

📋 Executive Summary

This roadmap outlines the week-by-week implementation plan to achieve:

- **Week 4**: First autonomous survey ($10-20/week)
- **Week 8**: Quenito at $500/week
- **Week 12**: Quenito + Quenita at $2,000/week

Each week includes specific technical tasks, testing milestones, and revenue targets.

🗓️ Month 1: Foundation & First Automation

Week 1: Platform Adapter Foundation

Goal: MyOpinions adapter + first manual learning surveys

Technical Tasks:

1. **Create MyOpinions Platform Adapter**

```python
```

```python
# platform_adapters/adapters/myopinions_adapter.py
from typing import List, Dict, Optional
from datetime import datetime
from platform_adapters.base_adapter import BasePlatformAdapter

class MyOpinionsAdapter(BasePlatformAdapter):
    """MyOpinions.com.au platform adapter"""

    def __init__(self, browser_manager):
        super().__init__(browser_manager)
        self.platform_name = "myopinions"
        self.base_url = "https://www.myopinions.com.au"
        self.points_per_dollar = 100  # 2000 points = $20 AUD

        # Platform-specific selectors
        self.selectors = {
            "dashboard": "div.dashboard-content",
            "survey_list": "div.survey-list-item",
            "points": "span.survey-points",
            "time": "span.survey-duration",
            "start_button": "button.start-survey",
            "bonus_indicator": "div.bonus-tier-badge"
        }

        # Bonus tier configuration
        self.bonus_tiers = {
            "starter": 0.00,
            "bronze": 0.05,   # 5% weekly bonus
            "silver": 0.075,  # 7.5% weekly bonus
            "gold": 0.10      # 10% weekly bonus
        }

    async def navigate_to_surveys(self) -> bool:
        """Navigate to survey dashboard"""
        try:
            await self.page.goto(f"{self.base_url}/auth/dashboard",
                                 wait_until="networkidle")

            # Verify we're logged in
            if "login" in self.page.url.lower():
                self.logger.error("Not logged in - cookie transfer may have failed")
                return False
            return True
        except Exception as e:
            self.logger.error(f"Navigation failed: {e}")
            return False

    async def get_available_surveys(self) -> List[Dict]:
        """Extract available surveys with metadata"""
        surveys = []

        try:
            # Wait for survey list to load
            await self.page.wait_for_selector(self.selectors["survey_list"])

            # Extract all survey items
            survey_elements = await self.page.query_selector_all(
                self.selectors["survey_list"]
            )

            for element in survey_elements:
                # Extract survey details
                points_text = await element.query_selector(
                    self.selectors["points"]
                ).inner_text()
                time_text = await element.query_selector(
                    self.selectors["time"]
                ).inner_text()

                # Parse points (e.g., "250 points" -> 250)
                points = int(''.join(filter(str.isdigit, points_text)))

                # Parse time (e.g., "15 mins" -> 15)
                time_minutes = int(''.join(filter(str.isdigit, time_text)))

                # Calculate value
                dollar_value = points / self.points_per_dollar
                hourly_rate = (dollar_value / time_minutes) * 60

                surveys.append({
                    "element": element,
                    "points": points,
                    "time_minutes": time_minutes,
                    "dollar_value": dollar_value,
                    "hourly_rate": hourly_rate,
                    "title": await element.query_selector("h3").inner_text()
                })

        except Exception as e:
            self.logger.error(f"Failed to extract surveys: {e}")

        return surveys

    async def select_best_survey(self, surveys: List[Dict]) -> Optional[Dict]:
        """Select optimal survey based on value and time"""
        if not surveys:
            return None

        # Sort by hourly rate, then by total value
        sorted_surveys = sorted(
            surveys,
            key=lambda x: (x["hourly_rate"], x["dollar_value"]),
            reverse=True
        )

        # Additional filtering logic
        for survey in sorted_surveys:
            # Skip very long surveys initially
            if survey["time_minutes"] > 45:
                continue

            # Skip very low value surveys
            if survey["dollar_value"] < 1.00:
                continue

            return survey

        # If no ideal survey, take the best available
        return sorted_surveys[0] if sorted_surveys else None

    async def start_survey(self, survey: Dict) -> bool:
        """Click start button for selected survey"""
        try:
            start_button = await survey["element"].query_selector(
                self.selectors["start_button"]
            )

            # Human-like delay before clicking
            await self.page.wait_for_timeout(
                random.randint(1000, 3000)
        )

        await start_button.click()

        # Wait for survey to load
        await self.page.wait_for_load_state("networkidle")

        return True

      except Exception as e:
        self.logger.error(f"Failed to start survey: {e}")
        return False

    async def track_bonus_progress(self) -> Dict:
      """Monitor bonus tier and weekly earnings"""
      try:
        # Extract current tier
        tier_element = await self.page.query_selector(
          self.selectors["bonus_indicator"]
        )
        current_tier = await tier_element.inner_text()

        # Extract weekly points (you'd need to navigate to a stats page)
        # This is simplified - real implementation would be more complex

        return {
          "current_tier": current_tier.lower(),
          "bonus_rate": self.bonus_tiers.get(current_tier.lower(), 0),
          "next_tier_requirement": self._get_next_tier_requirement(current_tier)
        }

      except Exception as e:
        self.logger.error(f"Failed to track bonus: {e}")
        return {}
```

2. **Setup Quenito Profile Structure**

```python
```

```
# personas/quenito/profile.json
{
 "identity": {
  "first_name": "Jack",
  "last_name": "Chen",
  "display_name": "Quenito",
  "date_of_birth": "1991-03-15",
  "age": 34,
  "gender": "male",
  "email": "quenito.surveys@gmail.com"
 },

 "demographics": {
  "location": {
   "suburb": "Sydney",
   "state": "NSW",
   "postcode": "2000",
   "country": "Australia",
   "timezone": "Australia/Sydney"
  },
  "household": {
   "status": "married",
   "children": 1,
   "household_size": 3,
   "household_income": "$75,000-$100,000"
  },
  "employment": {
   "status": "full_time",
   "occupation": "Software Developer",
   "industry": "Technology",
   "education": "Bachelor's Degree"
  }
 },

 "psychographics": {
  "interests": [
   "technology", "gaming", "fitness",
   "cooking", "travel", "photography"
  ],
  "shopping_behavior": {
   "online_frequency": "weekly",
   "preferred_retailers": ["Amazon", "eBay", "JB Hi-Fi"],
   "price_sensitivity": "moderate",
      "brand_loyalty": "moderate"
    },
    "media_consumption": {
      "social_media": ["LinkedIn", "Reddit", "Twitter"],
      "news_sources": ["ABC News", "The Guardian", "TechCrunch"],
      "streaming": ["Netflix", "YouTube", "Spotify"]
    },
    "values": [
      "innovation", "efficiency", "family",
      "health", "continuous_learning"
    ]
  },

  "response_patterns": {
    "style": "analytical_thoughtful",
    "consistency_markers": {
      "brand_preferences": "technology_focused",
      "price_ranges": "mid_to_premium",
      "quality_over_price": true
    },
    "timing": {
      "reading_speed_wpm": 250,
      "thinking_pause_ms": [1500, 3000],
      "typing_speed_cpm": 280
    },
    "likert_tendency": {
      "default": "slightly_positive",
      "avoid_extremes": true,
      "use_full_scale": true
    }
  },

  "platform_profiles": {
    "myopinions": {
      "username": "jchen91_syd",
      "member_since": "2025-08-01",
      "current_tier": "bronze",
      "lifetime_points": 2450,
      "surveys_completed": 12
    }
  }
}
```

Testing Milestones:

- [ ] Successfully connect to MyOpinions with cookie transfer
- [ ] Extract survey list and display available options
- [ ] Complete 5 surveys manually while logging patterns
- [ ] Verify Quenito profile data consistency

Revenue Target: $20-40 (manual completion)

Week 2: Handler Integration & Learning Loop

Goal: Connect handlers to platform adapter, enable learning mode

Technical Tasks:

1. **Integrate Existing Handlers with Platform Adapter**
2. **Enhanced Learning Loop for Manual Intervention**
3. **Create Survey Session Manager**

```python
```

```python
# core/session_manager.py
class SurveySessionManager:
  """Manages end-to-end survey sessions"""

  def __init__(self, persona_name: str, platform_name: str):
    self.persona = PersonaManager(persona_name)
    self.platform = self._get_platform_adapter(platform_name)
    self.handlers = HandlerFactory()
    self.brain = KnowledgeBase()
    self.learning_mode = True

  async def run_survey_session(self):
    """Complete survey flow"""
    # 1. Navigate to platform
    await self.platform.navigate_to_surveys()

    # 2. Select best survey
    surveys = await self.platform.get_available_surveys()
    best_survey = await self.platform.select_best_survey(surveys)

    # 3. Start survey
    await self.platform.start_survey(best_survey)

    # 4. Process questions with handlers
    while not await self._is_survey_complete():
      question = await self._detect_current_question()
      handler = self.handlers.get_handler(question)

      if self.learning_mode and handler.confidence < 0.8:
        # Request manual intervention
        response = await self._manual_intervention(question)
        # Learn from response
        await self.brain.learn_pattern(question, response)
      else:
        # Automated response
        await handler.handle_question(question)
```

Testing Milestones:

- [ ] Complete survey with 50% automation
- [ ] Learning loop captures new patterns
- [ ] Session recovery works after interruption

Revenue Target: $40-60

Week 3: Automation Refinement

Goal: Achieve 80%+ automation on MyOpinions

Technical Tasks:

1. **Optimize Handler Confidence**
2. **Implement Monitoring Dashboard**

python

```python
```

```python
# monitoring/dashboard.py
class QuentioMonitoringDashboard:
    """Real-time monitoring and analytics"""

    def __init__(self):
        self.metrics = {
            "automation_rate": AutomationTracker(),
            "earnings": EarningsTracker(),
            "platform_health": PlatformHealthMonitor(),
            "learning_progress": LearningProgressTracker()
        }

    def generate_daily_report(self) -> Dict:
        """Generate comprehensive daily metrics"""
        return {
            "date": datetime.now().isoformat(),
            "automation_metrics": {
                "overall_rate": self.metrics["automation_rate"].get_rate(),
                "by_handler": self.metrics["automation_rate"].get_by_handler(),
                "improvement": self.metrics["automation_rate"].get_trend()
            },
            "earnings": {
                "daily_total": self.metrics["earnings"].get_daily(),
                "weekly_projection": self.metrics["earnings"].project_weekly(),
                "by_platform": self.metrics["earnings"].get_by_platform()
            },
            "platform_status": {
                "account_health": self.metrics["platform_health"].check_all(),
                "screen_out_rate": self.metrics["platform_health"].get_screen_outs(),
                "warnings": self.metrics["platform_health"].get_warnings()
            },
            "learning": {
                "new_patterns": self.metrics["learning_progress"].get_new_patterns(),
                "confidence_improvements": self.metrics["learning_progress"].get_improvements()
            }
        }

    def create_visual_dashboard(self):
        """Generate HTML dashboard with charts"""
        # Creates beautiful dashboard with:
        # - Automation rate over time
        # - Earnings progression
        # - Platform health indicators
        # - Learning metrics
```

Testing Milestones:

- [ ] 80%+ automation rate achieved
- [ ] Complete 5-8 surveys daily
- [ ] Dashboard shows real-time metrics

Revenue Target: $80-120

Week 4: Multi-Platform Expansion

Goal: Add OpinionWorld and LifePointsPanel

Technical Tasks:

1. **Create Platform Adapters for OpinionWorld & LifePointsPanel**
2. **Implement Platform Rotation Logic**

```python
```

```
# scheduling/rotation_scheduler.py
class PlatformRotationScheduler:
    """Manages daily platform assignments"""

    def __init__(self):
        self.platforms = {
            "tier1": ["myopinions", "opinionworld", "lifepointspanel"],
            "tier2": ["primeopinion", "octopus"]
        }

    def generate_weekly_schedule(self, personas: List[str]) -> Dict:
        """Create non-overlapping schedule"""
        schedule = {}

        # Example week
        week_template = {
            "monday": {
                "quenito": ["myopinions", "lifepointspanel"],
                "quenita": ["opinionworld", "primeopinion"]
            },
            "tuesday": {
                "quenito": ["opinionworld", "octopus"],
                "quenita": ["myopinions", "lifepointspanel"]
            },
            "wednesday": "REST_DAY",
            "thursday": {
                "quenito": ["primeopinion", "myopinions"],
                "quenita": ["lifepointspanel", "octopus"]
            },
            # ... continue pattern
        }

        return self._validate_no_conflicts(week_template)
```

Testing Milestones:

- [ ] 3 platforms fully automated
- [ ] Platform rotation working smoothly
- [ ] Daily earnings reach $20-30

Revenue Target: $120-150

🗓️ Month 2: Scaling & Quenita Introduction

Week 5-6: Platform Optimization

Goal: Maximize earnings on existing platforms

Technical Tasks:

1. **Bonus System Optimization**
2. **Survey Selection Algorithm Enhancement**
3. **Advanced Pattern Learning**

Testing Milestones:

- [ ] Reach higher bonus tiers
- [ ] Optimal survey selection working
- [ ] Daily earnings: $30-40

Revenue Target: $200-250/week

Week 7-8: Quenita Launch

Goal: Introduce Quenita with manual learning phase

Technical Tasks:

1. **Create Quenita Profile**

python

```python
```

```
# personas/quenita/profile.json
{
 "identity": {
  "first_name": "Emma",
  "last_name": "Rodriguez",
  "display_name": "Quenita",
  "date_of_birth": "1993-07-22",
  "age": 32,
  "gender": "female",
  "email": "quenita.surveys@gmail.com"
 },

 "demographics": {
  "location": {
   "suburb": "Parramatta",
   "state": "NSW",
   "postcode": "2150",
   "country": "Australia"
  },
  "household": {
   "status": "married",
   "children": 2,
   "household_size": 4,
   "household_income": "$100,000-$125,000"
  },
  "employment": {
   "status": "full_time",
   "occupation": "Marketing Manager",
   "industry": "Retail",
   "education": "Master's Degree"
  }
 },

 "psychographics": {
  "interests": [
   "fashion", "wellness", "parenting",
   "home_decor", "sustainable_living", "yoga"
  ],
  "shopping_behavior": {
   "online_frequency": "multiple_weekly",
   "preferred_retailers": ["Myer", "David Jones", "The Iconic"],
   "price_sensitivity": "value_conscious",
   "brand_loyalty": "high"
```

```
    },
    "values": [
      "family", "sustainability", "work_life_balance",
      "health", "community"
    ]
  },

  "response_patterns": {
    "style": "practical_experiential",
    "timing": {
      "reading_speed_wpm": 220,
      "thinking_pause_ms": [2000, 4000],
      "typing_speed_cpm": 250
    }
  }
}
```

2. **Separate Browser Profiles**
3. **Learning Transfer System**

Testing Milestones:

- [ ] Quenita profile created and consistent
- [ ] Separate browser sessions working
- [ ] Combined daily earnings: $50-60

Revenue Target: $400-500/week (combined)

📅 Month 3: Full Automation & Optimization

Week 9-10: Dual Persona Optimization

Goal: Both personas running at high automation

Technical Tasks:

1. **Cross-Persona Learning Benefits**
2. **Advanced Scheduling Optimization**
3. **Platform Expansion (add PrimeOpinion)**

Testing Milestones:

- [ ] Both personas 90%+ automated

- [ ] 4-5 platforms each
- [ ] Daily earnings: $100-150

Revenue Target: $700-1000/week

Week 11-12: Target Achievement

Goal: Reach and stabilize at $2,000/week

Technical Tasks:

1. **Fine-tune All Systems**
2. **Add Remaining Platforms**
3. **Implement Future Scaling Prep**

```python
# analytics/revenue_tracker.py
class RevenueOptimizer:
    """Maximizes revenue across all personas and platforms"""

    def optimize_daily_strategy(self) -> Dict:
        """Dynamic optimization based on platform availability"""
        return {
            "platform_priorities": self._rank_platforms_by_value(),
            "time_allocation": self._optimize_time_slots(),
            "bonus_opportunities": self._identify_bonus_targets(),
            "risk_mitigation": self._check_account_health()
        }
```

Testing Milestones:

- [ ] Consistent $2,000/week achieved
- [ ] All systems stable and optimized
- [ ] Ready for Phase 2 expansion

Revenue Target: $2,000/week 🎉

📊 Success Metrics Dashboard

Weekly Tracking Template

```
Week X Summary:
├── Automation Rate: XX%
├── Surveys Completed:
│   ├── Quenito: XX
│   └── Quenita: XX
├── Earnings:
│   ├── Daily Average: $XX
│   ├── Weekly Total: $XXX
│   └── Projection: $X,XXX/month
├── Platform Health:
│   ├── Screen-out Rate: X%
│   ├── Account Warnings: None
│   └── Bonus Tiers: [Status]
└── Next Week Focus: [Key objectives]
```

🎯 Critical Success Factors

1. **Week 1-2**: Foundation MUST be solid
2. **Week 3-4**: Automation rate is key metric
3. **Week 5-8**: Quenita launch timing critical
4. **Week 9-12**: Optimization and stability focus

🚨 Risk Mitigation Checkpoints

- **Week 2**: Verify no account flags
- **Week 4**: Check screen-out rates < 10%
- **Week 6**: Platform health assessment
- **Week 8**: Quenita account establishment
- **Week 10**: Full system audit
- **Week 12**: Long-term sustainability check

🎉 Celebration Milestones

- **First Autonomous Survey**: Pizza night! 🍕
- **$100/week**: Team dinner 🍽️
- **$500/week**: Weekend getaway planning 🏖️
- **$1000/week**: Major celebration 🎊

- **$2000/week**: Phase 2 planning retreat! 🚀

"From $0 to $2,000/week in 12 weeks - Quenito and Quenita are ready to scale beyond physical limitations!"