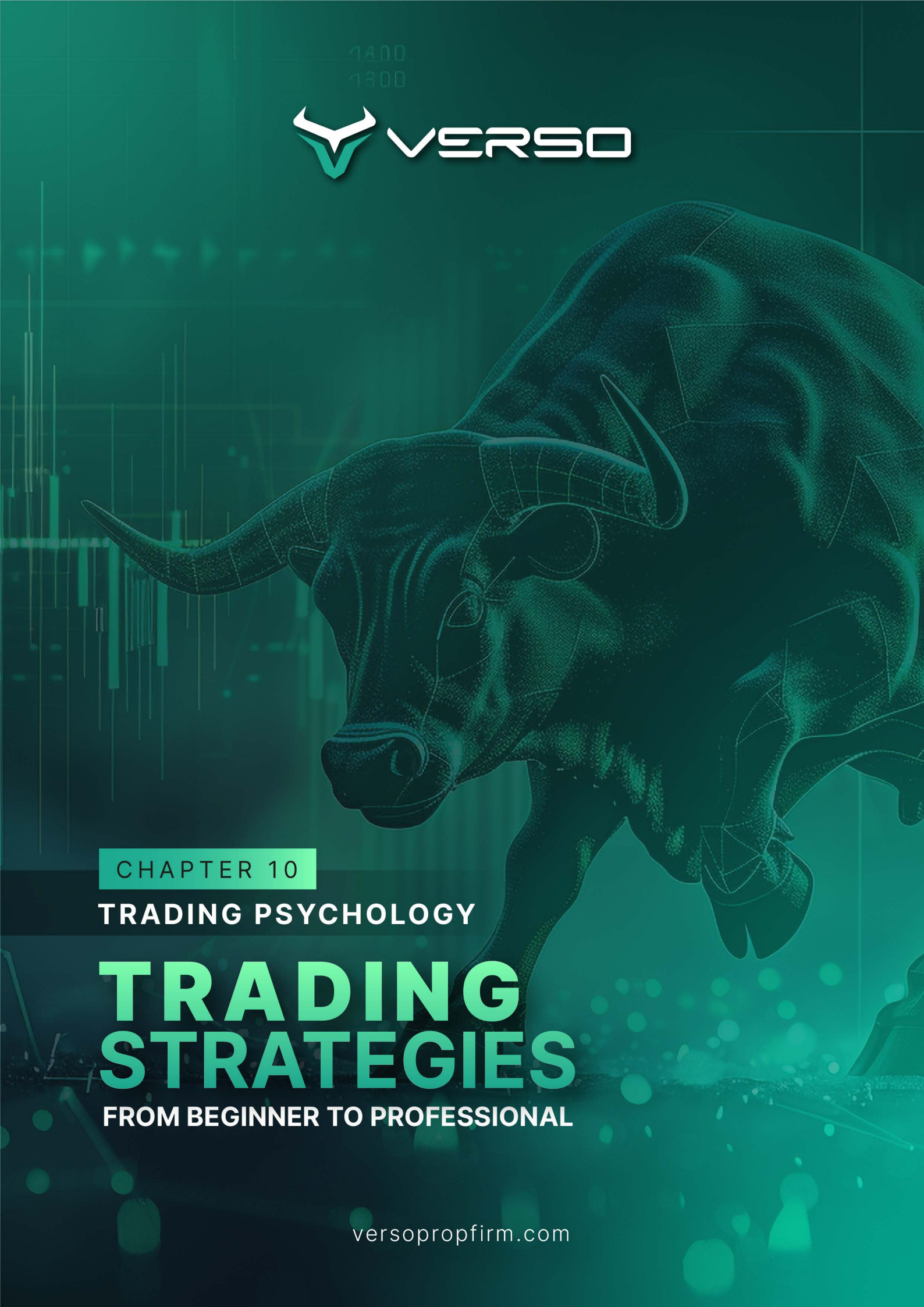

CHAPTER 10

TRADING PSYCHOLOGY

TRADING STRATEGIES

FROM BEGINNER TO PROFESSIONAL

versopropfirm.com

Trading Psychology

Trading psychology is the most critical and least understood battlefield in the financial world. You can have the best strategy in the world, but if your mind isn't prepared, the market will crush you without mercy. As legendary trader Jesse Livermore said: "Wall Street never changes, the markets go up and down, but human nature never changes."

He Trader's Internal Enemies

Fear

Fear paralyzes. It makes you exit winning positions too early and hold onto losing positions hoping they'll "recover."

Greed

Greed blinds you. It makes you chase every opportunity, over-leverage positions, and never take profits.

Hope

Unfounded hope is deadly. "Hoping" a losing position will recover is a recipe for disaster.

Revenge Trading

After a loss, the desire to "recover" money immediately leads to irrational decisions.

The MIND Framework for Mental Control
M - Mindfulness
I - Identification (of Emotions)
N – Neutralization
D - Decision (Data-Based)

```
python
class TradingPsychologyTracker:
    def __init__(self):
        self.emotional_states = []
        self.trade_journal = []

    def log_pre_trade_state(self, trade_setup):
        """
        Logs emotional state before the trade
        """
        emotional_assessment = {
            'timestamp': datetime.now(),
            'confidence_level': self.rate_confidence(1, 10),
            'stress_level': self.rate_stress(1, 10),
            'last_trades_impact': self.assess_recent_performance(),
            'market_fear_greed': self.external_sentiment(),
            'physical_state': self.assess_physical_condition(),
            'trade_setup_quality': trade_setup['quality_score']
        }

        # Determines if trading is advisable based on mental state
        should_trade = self.mental_state_check(emotional_assessment)
        return {
            'assessment': emotional_assessment,
            'trade_recommended': should_trade,
            'risk_adjustment': self.calculate_risk_adjustment(emotional_assessment)
        }

    def mental_state_check(self, assessment):
        """
        Determines if mental state is suitable for trading
        """
        red_flags = 0

        # High Stress
        if assessment['stress_level'] > 7:
            red_flags += 2
```

```
    # Very low or very high confidence
        if assessment['confidence_level'] < 3 or assessment['confidence_level'] > 8:
            red_flags += 1
```

```
# Negative impact from recent trades
if assessment['last_trades_impact']<-2: red_flags +=2

# Poor physical state
if assessment['physical_state']<4:
red_flags +=1

# Do not trade if there are 3+ red flags
return red_flags <3
def calculate_risk_adjustment(self, assessment):
"""
Adjusts risk based on psychological state

"""
base_risk =1.0

# Reduce risk if stress is high
if assessment['stress_level']>6: base_risk *=0.7

# Reduce risk if confidence is very low
    if assessment['confidence_level']<4:
base_risk *=0.8

# Reduce risk after losses
if assessment['last_trades_impact']<-1: base_risk *=0.6

# Never increase risk due to overconfidence
returnmin(base_risk,1.0)
```

Emotional Control Techniques

1. Pre-Market Routine

```
classPreMarketRoutine:

def__init__(self):
self.routine_checklist =[
'market_analysis_complete',
'trading_plan_written',
'risk_parameters_set',
'mental_state_assessed',
'physical_readiness_confirmed'
]
def execute_routine(self):
"""
 Executes complete pre-market routine
 """
 routine_results ={}


# 1. Market analysis (30 minutes)
routine_results['market_analysis']= self.market_analysis()

# 2. Daily trading plan
routine_results['daily_plan']= self.create_daily_plan()

# 3. Risk parameters
routine_results['risk_params']= self.set_risk_parameters()

# 4. Mental state
routine_results['mental_state']= self.mental_preparation()

# 5. Physical preparation
routine_results['physical_prep']= self.physical_preparation()

# Final decision: Ready to trade?
readiness_score = self.calculate_readiness(routine_results)

return{
'ready_to_trade': readiness_score >0.8,
'readiness_score': readiness_score,
```

```
'routine_results': routine_results}
defmental_preparation(self):
"""

Specific mental preparation techniques
 """
techniques ={
'meditation': self.brief_meditation(5),# 5 minutos
'visualization': self.trade_visualization(),
'affirmations': self.positive_affirmations(),
'goal_review': self.review_trading_goals()
}
return techniques
defbrief_meditation(self, minutes):
"""
 Guide for brief meditation
 """
return{
'technique':'breath_focus',
'duration': minutes,
'steps':[
    '1. Sit comfortably, back straight',
    '2. Close your eyes, breathe deeply',
    '3. Count breaths from 1 to 10, repeat',
    '4. If your mind wanders, return to breathing',
    '5. Finish with 3 deep breaths']
}
def trade_visualization(self):
"""
Visualization of successful trades
"""
return{
'scenario':'perfect_trade_execution',
'steps':[
'Visualize identifying the perfect setup',
'See yourself entering with confidence',
'Imagine managing the position calmly',
'Visualize exiting with planned profit',
'Feel the satisfaction of a disciplined trade'
]
}
```

2. In-Trade Emotional Management

```
class InTradeEmotionalManager:
def__init__(self):
self.emotional_triggers ={
'position_moving_against': self.handle_adverse_move,
'position_in_profit': self.handle_profit_euphoria,
'high_volatility': self.handle_volatility_stress,
'news_impact': self.handle_news_anxiety}
def handle_adverse_move(self, position, unrealized_pnl):
"""
Handles emotions when position moves against
"""
response_protocol =[]

# 1. Controlled breathing
response_protocol.append({
'action':'breathing_exercise',
'technique':'4-7-8 breathing',
'repetitions':3})

# 2. Review original plan
response_protocol.append({
'action':'plan_review',
'questions':[
'Is my original thesis still valid?',
'Am I within my stop loss?',
'What does my trading plan say?'
]
})

# 3. Rule-based decision, not emotion-driven
ifabs(unrealized_pnl)>= position['stop_loss_amount']:
response_protocol.append({
'action':'exit_position',
'reason':'stop_loss_triggered',
'emotion_override':True})
return response_protocol
def handle_profit_euphoria(self, position, unrealized_pnl):
"""
```

```
Handles euphoria when in profit
"""
 response_protocol =[]

# Overconfidence alert
if unrealized_pnl > position['expected_profit']*2: response_protocol.append({
'alert':'exceptional_profit_warning',
'message':'Ganancia excepcional - mantén disciplina',
'actions':[
'Review original take-profit',
'Consider partial profit-taking',
'DO NOT increase next position size'
]})
return response_protocol
def
real_time_emotional_check(self):
"""
Quick emotional state check during trading
"""
questions =[
    "Am I following my plan? (Yes/No)",
    "Do I feel anxious or euphoric? (1-10)",
    "Am I thinking about revenge trading? (Yes/No)",
    "Is my sizing appropriate? (Yes/No)"

# In real implementation, this would be a prompt to the trader
return questions
```

Long-Term Discipline Development
The Psychological Trading Journal

```
classPsychologyTradingJournal:
def__init__(self):
 self.entries =[]
def create_trade_entry(self, trade_data):
"""
 Creates a complete journal entry including psychological aspects
 """
entry ={
'timestamp': datetime.now(),
    'trade_details': trade_data,
```

```
    'pre_trade_psychology': {
        'confidence_level': self.rate_scale(1, 10, "How confident did you feel?"),
        'fear_level': self.rate_scale(1, 10, "How fearful did you feel?"),
        'external_pressures': self.list_pressures(),
        'mental_clarity': self.rate_scale(1, 10, "How clearly were you thinking?")},
'during_trade_psychology':{
'emotional_changes': self.track_emotional_journey(),
'decision_points': self.identify_key_decisions(),
'stress_moments': self.identify_stress_peaks()

},
'post_trade_psychology':{
'satisfaction_level': self.rate_scale(1,10,"¿Qué tan satisfecho estás?"),
'regrets': self.identify_regrets(),
'lessons_learned': self.extract_lessons(),
'emotional_state': self.assess_post_trade_emotion()
}
} self.entries.append(entry)
return entry
defanalyze_psychological_patterns(self, lookback_trades=50):
"""
Analyzes psychological patterns in recent trades
"""
recent_entries = self.entries[-lookback_trades:]
patterns ={
'confidence_vs_performance': self.correlation_analysis(
[e['pre_trade_psychology']['confidence_level']for e in recent_entries],
[e['trade_details']['pnl']for e in recent_entries]),
'fear_impact': self.analyze_fear_impact(recent_entries),
'common_mistakes': self.identify_recurring_mistakes(recent_entries),
'emotional_triggers': self.identify_triggers(recent_entries)}
return patterns
def generate_psychological_recommendations(self, patterns):
"""

Generates recommendations based on identified patterns
 """
recommendations =[]
# Confidence analysis
if patterns['confidence_vs_performance']['correlation']<-0.3:
recommendations.append({
```

```
'area':'overconfidence',
'issue':'
'issue': 'Higher confidence correlates with worse performance',
    'solution': 'Implement checks when confidence > 8/10'
})

# Fear analysis
if patterns['fear_impact']['high_fear_success_rate']>
patterns['fear_impact']['low_fear_success_rate'
recommendations.append({
'area': 'fear_management',
'issue': 'Trades with more fear are more successful',
'solution': 'Fear may indicate better risk analysis'
})
return recommendations
```

Advanced Mental Control Techniques

1. Mindfulness Trading

Python

```
classMindfulnessTrading:
def__init__(self):
self.mindfulness_techniques ={
'present_moment_awareness': self.present_moment_exercise,
'emotional_labeling': self.emotion_labeling_technique,
'body_scan': self.trading_body_scan,
'mindful_decision_making': self.mindful_decision_process
}
def present_moment_exercise(self):
"""
Technique to maintain present-moment awareness
"""
return{
'name':'Present Moment Awareness',
'duration':'2-3 minutos','steps':[
'Pause what you're doing',
'Note 3 things you can see on your screen',
'Note 2 sounds you can hear',
```

```
'Note 1 physical sensation you feel',
'Take a deep breath and return to trading'
],
'when_to_use':'Antes de cada trade importante'
}
def emotion_labeling_technique(self):
"""
Technique to identify and label emotions
"""
return{
'name':'Emotion Labeling',
'process':[
'Pause when you feel intense emotion',
'Ask yourself: "What am I feeling exactly?"',
'Label the emotion: "I am feeling anxiety"',
'Ask: "Is this emotion helping my trading?"',
'If not helpful, take 5 deep breaths'
],
'emotions_to_watch':[
'FOMO (Fear of Missing Out)',
'Revenge (Deseo de venganza)',
'Euphoria (Euforia)',
'Panic (Pánico)',
'Frustration (Frustración)'
]
}
```

2. Cognitive Behavioral Techniques for Trading

Python

```
classCBTForTrading:
def__init__(self):
self.cognitive_distortions ={
'all_or_nothing': 'Thinking one losing trade makes you a bad trader',
'catastrophizing': 'Imagining the worst possible scenarios',
'mental_filter': 'Focusing only on losing trades',
'emotional_reasoning': 'Feeling something is wrong = it must be wrong',
'fortune_telling': 'Predicting the future without evidence'
}
def challenge_negative_thought(self, negative_thought):
```

```
"""
Challenges negative thoughts using CBT
"""
 challenge_questions =[
"Is this thought based on facts or emotions?",
"What evidence supports this thought?",
"What evidence contradicts this thought?",
"What would I say to a friend with this thought?",
"What is a more balanced perspective?",
"How does this thought help or harm me?"
]
return{
'original_thought': negative_thought,
'challenge_questions': challenge_questions,
'reframe_process': self.thought_reframing_process()
}
defthought_reframing_process(self):

"""
Process to reframe negative thoughts
"""
return{
'steps':[
'Identify the automatic negative thought',
'Determine which cognitive distortion it represents',
'Look for evidence for and against it',
'Create a more balanced and realistic thought',
'Act based on the reframed thought'
],
'example':{
'negative': '"I lost money, I'm terrible at trading"',
'distortion': 'all_or_nothing + mental_filter',
'balanced': '"I lost on this trade, but it's part of the process. My overall stats are still
good"
'}
}
```

Building Anti-Stress Routines
Daily Trading Routine Optimization

```
classOptimalTradingRoutine:
```

```
def__init__(self):
self.morning_routine = self.create_morning_routine()
self.trading_session_routine = self.create_session_routine()
self.evening_routine = self.create_evening_routine()
def create_morning_routine(self):
"""
 Optimized morning routine for performance
"""
return{
'duration':'60-90 minutos antes de mercado','activities':[
{
'time':'7:00-7:15',
'activity': 'Light exercise or walk',
'purpose': 'Activate body and mind"
},
{'time':'7:15-7:25',
'activity':'Meditación o mindfulness',
'purpose': Center the mind
}
,{
'time':'7:25-7:40',
'activity': 'Healthy breakfast',
'purpose': 'Sustainable energy'
'},
{
'time':'7:40-8:15',
'activity': 'Market analysis',
'purpose': 'Technical preparation'
},
{'time':'8:15-8:30',
'activity': 'Daily plan and visualization',
'purpose': 'Psychological preparation'
}
]
}
def create_session_routine(self):
"""
Routine during trading session
"""
return{
'every_hour':[
'Check estado emocional (1-10 scale)',
```

'Hidratación (vaso de agua)',
'Postura check y estiramiento'
],
'after_each_trade':[
'Immediate emotional log',
'Ask: Did I follow my plan?',
'Deep breathing'
],
'mid_session_break':{
'timing':'Después de 2-3 horas',
'duration':'15-20 minutos',
'activities':[
'Step away from screens',
'Short walk',
'Healthy snack',
'Quick performance review'

]
}
}

versopropfirm.com