title: “SKILLS ASSESSMENT GUIDE”
author: “JobGate Career Quest Team”
date: “\today”
geometry: margin=1in
fontfamily: lmodern
fontsize: 11pt
documentclass: article
header-includes:
- \usepackage[utf8]{inputenc}
- \usepackage[T1]{fontenc}
- \usepackage{lmodern}
- \usepackage{microtype}
- \usepackage{xcolor}
- \usepackage{hyperref}
- \hypersetup{colorlinks=true, linkcolor=blue, urlcolor=blue}
output:
pdf_document:
engine: pdflatex
latex_engine: pdflatex

SKILLS ASSESSMENT GUIDE

Detailed Specifications for All Assessment Types

ASSESSMENT OVERVIEW

This guide provides comprehensive specifications for implementing all assessment types in the JobGate Career Quest platform. Each assessment includes detailed question examples, scoring methodologies, and implementation requirements.

Assessment Categories

1. **Cognitive Assessments (6 types):** Numerical, Verbal, Logical, Abstract, Diagrammatic, Spatial
2. **Personality & Behavioral (2 types):** Big Five Personality, Situational Judgment Tests
3. **Technical Skills (Expandable):** Software Engineering, Marketing, Industry-specific

---

COGNITIVE ASSESSMENTS

1.1 NUMERICAL REASONING ASSESSMENT

Purpose: Evaluate candidate’s ability to work with numerical data, perform calculations, and interpret quantitative information.

Objective: Measure mathematical reasoning, data interpretation, and numerical problem-solving skills essential for roles requiring analytical thinking.

Test Structure

- **Duration:** 20 minutes
- **Questions:** 25 questions
- **Question Distribution:**
- Basic arithmetic and percentages (20%)

- Ratio and proportion problems (20%)
- Data interpretation from charts/graphs (30%)
- Financial calculations (20%)
- Statistical reasoning (10%)

Difficulty Levels

- **Easy (40% - 10 questions):** Basic arithmetic, simple percentages
- **Medium (45% - 11 questions):** Data interpretation, financial calculations
- **Hard (15% - 4 questions):** Complex statistical reasoning, multi-step problems

Sample Questions   Easy Level - Basic Arithmetic:

```
{
 "question_id": "NUM_001",
 "difficulty": "easy",
 "question": "A product originally costs $120. After a 15% discount, what is the new price?",
 "options": ["$102", "$108", "$105", "$110"],
 "correct_answer": "$102",
 "explanation": "15% of $120 = $18. New price = $120 - $18 = $102",
 "time_limit": 45,
 "points": 1
}
```

Medium Level - Data Interpretation:

```
{
 "question_id": "NUM_015",
 "difficulty": "medium",
 "question": "Based on the sales chart below, what was the percentage increase in sales from Q1 to Q2?",
 "chart_data": {
 "type": "bar_chart",
 "data": {"Q1": 15000, "Q2": 18000, "Q3": 22000, "Q4": 19000},
 "title": "Quarterly Sales (USD)"
 },
 "options": ["15%", "20%", "25%", "30%"],
 "correct_answer": "20%",
 "explanation": "Increase = (18000-15000)/15000 × 100 = 3000/15000 × 100 = 20%",
 "time_limit": 60,
 "points": 2
}
```

Hard Level - Statistical Reasoning:

```
{
 "question_id": "NUM_023",
 "difficulty": "hard",
 "question": "A company's revenue follows a normal distribution with mean $500K and standard deviation $50K
 "options": ["68%", "95%", "75%", "85%"],
 "correct_answer": "68%",
 "explanation": "In a normal distribution, 68% of values fall within 1 standard deviation of the mean. $450
 "time_limit": 90,
 "points": 3
}
```

Scoring Algorithm

```
def calculate_numerical_score(responses, questions):
 raw_score = 0
```

```
time_bonus = 0

for response in responses:
question = questions[response.question_id]
if response.is_correct:
raw_score += question.points

# Time bonus for faster completion
time_used = response.time_spent
time_limit = question.time_limit
if time_used < time_limit * 0.7:
time_bonus += question.points * 0.1

total_possible = sum(q.points for q in questions)
percentage_score = (raw_score / total_possible) * 100
final_score = min(100, percentage_score + time_bonus)

return {
'raw_score': raw_score,
'percentage': percentage_score,
'time_bonus': time_bonus,
'final_score': final_score,
'percentile': calculate_percentile(final_score, 'numerical')
}
```

Implementation Requirements

- Chart/graph rendering system for data interpretation questions
- Timer with visual countdown
- Calculator functionality (basic operations only)
- Progress indicator
- Question randomization within difficulty levels
- Responsive design for mobile completion

1.2 VERBAL REASONING ASSESSMENT

Purpose: Assess candidate's ability to understand, analyze, and draw conclusions from written information.

Objective: Measure reading comprehension, vocabulary, logical deduction from text, and critical thinking skills.

Test Structure

- **Duration:** 25 minutes
- **Questions:** 30 questions
- **Question Distribution:**
- Reading comprehension passages (40% - 12 questions)
- Vocabulary in context (20% - 6 questions)
- Logical deduction from text (25% - 7 questions)
- Critical reasoning (15% - 5 questions)

Sample Questions **Reading Comprehension:**

```
{
 "question_id": "VER_001",
 "difficulty": "medium",
 "passage": "The rise of artificial intelligence in the workplace has sparked debate about job displacement
 "question": "According to the passage, what is the main advantage of AI in the workplace?",
 "options": [
 "Creative problem-solving abilities",
 "Emotional intelligence in customer relations",
 "Efficiency in routine and predictable tasks",
 "Strategic thinking and oversight"
 ],
 "correct_answer": "Efficiency in routine and predictable tasks",
 "explanation": "The passage states that AI 'excels at routine and predictable tasks' and companies report
 "time_limit": 90,
 "points": 2
}
```

Logical Deduction:

```
{
 "question_id": "VER_015",
 "difficulty": "medium",
 "question": "All successful entrepreneurs are risk-takers. Some risk-takers are innovators. Maria is a suc
 "options": [
 "Maria is an innovator",
 "Maria is a risk-taker",
 "All risk-takers are entrepreneurs",
 "Some entrepreneurs are innovators"
 ],
 "correct_answer": "Maria is a risk-taker",
 "explanation": "If all successful entrepreneurs are risk-takers, and Maria is a successful entrepreneur, t
 "time_limit": 75,
 "points": 2
}
```

Vocabulary in Context:

```
{
 "question_id": "VER_025",
 "difficulty": "easy",
 "question": "The CEO's decision to divest the underperforming division was met with approval from sharehol
 "options": [
 "To invest more money in",
 "To sell or dispose of",
 "To restructure completely",
 "To relocate to another country"
 ],
 "correct_answer": "To sell or dispose of",
 "explanation": "Divest means to sell off or dispose of business interests. The context of an 'underperform
 "time_limit": 45,
 "points": 1
}
```

Scoring Algorithm

```
def calculate_verbal_score(responses, questions):
 comprehension_score = 0
 vocabulary_score = 0
 reasoning_score = 0

 for response in responses:
 question = questions[response.question_id]
 if response.is_correct:
```

```python
 if question.category == 'comprehension':
 comprehension_score += question.points
 elif question.category == 'vocabulary':
 vocabulary_score += question.points
 elif question.category == 'reasoning':
 reasoning_score += question.points

 # Weighted scoring
 final_score = (
 comprehension_score * 0.4 +
 vocabulary_score * 0.2 +
 reasoning_score * 0.4
 )

 return {
 'comprehension': comprehension_score,
 'vocabulary': vocabulary_score,
 'reasoning': reasoning_score,
 'final_score': final_score,
 'percentile': calculate_percentile(final_score, 'verbal')
 }
```

1.3 LOGICAL REASONING ASSESSMENT

Purpose: Evaluate candidate's ability to identify patterns, relationships, and logical sequences.

Objective: Measure deductive and inductive reasoning, pattern recognition, and systematic problem-solving approaches.

Test Structure

- **Duration:** 20 minutes
- **Questions:** 20 questions
- **Question Distribution:**
- Pattern sequences (35% - 7 questions)
- Logical deduction problems (30% - 6 questions)
- Conditional reasoning (20% - 4 questions)
- Rule-based problems (15% - 3 questions)

Sample Questions Pattern Sequences:

```json
{
 "question_id": "LOG_001",
 "difficulty": "medium",
 "question": "What comes next in the sequence: 2, 6, 18, 54, ?",
 "options": ["108", "162", "216", "324"],
 "correct_answer": "162",
 "explanation": "Each number is multiplied by 3: 2×3=6, 6×3=18, 18×3=54, 54×3=162",
 "pattern_type": "multiplicative",
 "time_limit": 60,
 "points": 2
}
```

Conditional Reasoning:

```json
{
 "question_id": "LOG_012",
 "difficulty": "hard",
 "question": "If it rains, then the picnic is cancelled. The picnic was not cancelled. What can we conclude
 "options": [
 "It rained",
 "It did not rain",
 "The picnic happened",
 "Cannot determine from given information"
 ],
 "correct_answer": "It did not rain",
 "explanation": "This is modus tollens: If P then Q, not Q, therefore not P. Since the picnic wasn't cancel
 "logic_type": "modus_tollens",
 "time_limit": 75,
 "points": 3
}
```

Rule-Based Problem:

```
{
 "question_id": "LOG_018",
 "difficulty": "medium",
 "question": "In a certain code: BOOK = 5, TREE = 7, HOUSE = 9. What does COMPUTER equal?",
 "options": ["12", "15", "18", "21"],
 "correct_answer": "15",
 "explanation": "The code counts unique letters: BOOK(4 unique)=5, TREE(3 unique)=7, HOUSE(5 unique)=9. Pat
 "pattern_type": "letter_counting",
 "time_limit": 90,
 "points": 3
}
```

1.4 ABSTRACT REASONING ASSESSMENT

Purpose: Measure candidate’s ability to identify relationships and patterns in visual information without relying on language or numerical skills.

Objective: Assess fluid intelligence, visual pattern recognition, and non-verbal problem-solving abilities.

Test Structure

- **Duration:** 15 minutes
- **Questions:** 20 questions
- **Question Distribution:**
- Pattern matrices (40% - 8 questions)
- Shape sequences (30% - 6 questions)
- Transformation rules (20% - 4 questions)
- Odd-one-out (10% - 2 questions)

Sample Questions Pattern Matrix:

```
{
 "question_id": "ABS_001",
 "difficulty": "medium",
 "question": "Which shape completes the pattern?",
 "matrix": {
 "type": "3x3_grid",
 "pattern_description": "Each row contains circle, square, triangle. Each column has different orientations
 "missing_position": "bottom_right",
 "svg_data": "<!-- SVG pattern data -->"
 },
 "options": [
 {"svg": "<!-- Option A SVG -->", "id": "A"},
 {"svg": "<!-- Option B SVG -->", "id": "B"},
 {"svg": "<!-- Option C SVG -->", "id": "C"},
 {"svg": "<!-- Option D SVG -->", "id": "D"}
 ],
 "correct_answer": "C",
 "explanation": "The pattern follows shape progression in rows and rotation in columns.",
 "time_limit": 45,
 "points": 2
}
```

Shape Sequence:

```
{
 "question_id": "ABS_010",
 "difficulty": "hard",
 "question": "What comes next in the sequence?",
 "sequence": {
 "type": "shape_transformation",
 "transformations": ["rotate_90", "add_element", "change_color"],
 "svg_sequence": ["<!-- Shape 1 -->", "<!-- Shape 2 -->", "<!-- Shape 3 -->"]
 },
 "options": [
 {"svg": "<!-- Option A -->", "id": "A"},
 {"svg": "<!-- Option B -->", "id": "B"},
 {"svg": "<!-- Option C -->", "id": "C"},
 {"svg": "<!-- Option D -->", "id": "D"}
 ],
 "correct_answer": "B",
 "explanation": "Pattern involves 90° rotation + adding one element + color change progression.",
 "time_limit": 60,
 "points": 3
}
```

Implementation Requirements

- SVG-based pattern generation system
- Interactive drag-and-drop functionality
- Pattern validation algorithms
- Visual similarity detection
- Cultural fairness testing
- Progressive difficulty calibration

1.5 DIAGRAMMATIC REASONING ASSESSMENT

Purpose: Evaluate candidate's ability to work with flowcharts, process diagrams, and logical operators.

Objective: Measure process understanding, sequential logic, and ability to follow complex procedural instructions.

Test Structure

- **Duration:** 18 minutes
- **Questions:** 18 questions
- **Question Distribution:**
- Flowchart completion (40% - 7 questions)

- Process sequence identification (30% - 5 questions)
- Input-output transformations (20% - 4 questions)
- Decision tree navigation (10% - 2 questions)

Sample Questions Flowchart Completion:

```
{
 "question_id": "DIA_001",
 "difficulty": "medium",
 "question": "Complete the flowchart. If input is 15, what is the final output?",
 "flowchart": {
 "start": "input_number",
 "steps": [
 {"type": "decision", "condition": "number > 10", "yes": "multiply_by_2", "no": "add_5"},
 {"type": "process", "action": "multiply_by_2", "next": "subtract_3"},
 {"type": "process", "action": "add_5", "next": "subtract_3"},
 {"type": "process", "action": "subtract_3", "next": "output"}
 ],
 "missing_step": "subtract_3"
 },
 "options": ["27", "30", "25", "32"],
 "correct_answer": "27",
 "explanation": "15 > 10, so multiply by 2 = 30, then subtract 3 = 27",
 "time_limit": 75,
 "points": 2
}
```

Input-Output Transformation:

```
{
 "question_id": "DIA_012",
 "difficulty": "hard",
 "question": "Determine the pattern and find the output for input ABC123",
 "transformations": [
 {"input": "XYZ456", "output": "ZYX654"},
 {"input": "DEF789", "output": "FED987"},
 {"input": "GHI012", "output": "IHG210"}
 ],
 "options": ["CBA321", "ABC321", "CBA123", "321CBA"],
 "correct_answer": "CBA321",
 "explanation": "Pattern: reverse letters, reverse numbers. ABC123 CBA321",
 "time_limit": 90,
 "points": 3
}
```

1.6 SPATIAL REASONING ASSESSMENT

Purpose: Assess candidate’s ability to visualize and manipulate objects in three-dimensional space.

Objective: Measure spatial visualization, mental rotation abilities, and 3D problem-solving skills.

Test Structure

- **Duration:** 20 minutes
- **Questions:** 20 questions
- **Question Distribution:**
- 3D object rotation (35% - 7 questions)

- Shape folding/unfolding (25% - 5 questions)
- Cross-sections identification (20% - 4 questions)
- Perspective changes (20% - 4 questions)

Sample Questions **Mental Rotation:**

```
{
 "question_id": "SPA_001",
 "difficulty": "medium",
 "question": "Which option shows the same 3D object rotated 90° clockwise around the vertical axis?",
 "original_object": {
 "type": "3d_shape",
 "description": "L-shaped block with cube attachment",
 "svg_3d": "<!-- 3D SVG representation -->"
 },
 "options": [
 {"svg": "<!-- Rotated option A -->", "id": "A"},
 {"svg": "<!-- Rotated option B -->", "id": "B"},
 {"svg": "<!-- Rotated option C -->", "id": "C"},
 {"svg": "<!-- Rotated option D -->", "id": "D"}
 ],
 "correct_answer": "C",
 "explanation": "90° clockwise rotation moves the cube attachment to the right side.",
 "time_limit": 60,
 "points": 2
}
```

Paper Folding:

```
{
 "question_id": "SPA_015",
 "difficulty": "hard",
 "question": "A square paper is folded twice and holes are punched. How will it look when unfolded?",
 "folding_sequence": [
 {"step": 1, "action": "fold_diagonal", "svg": "<!-- Fold 1 -->"},
 {"step": 2, "action": "fold_half", "svg": "<!-- Fold 2 -->"},
 {"step": 3, "action": "punch_holes", "svg": "<!-- Punched -->"}
 ],
 "options": [
 {"svg": "<!-- Unfolded A -->", "id": "A"},
 {"svg": "<!-- Unfolded B -->", "id": "B"},
 {"svg": "<!-- Unfolded C -->", "id": "C"},
 {"svg": "<!-- Unfolded D -->", "id": "D"}
 ],
 "correct_answer": "B",
 "explanation": "Diagonal fold + half fold creates 8-way symmetry for hole punches.",
 "time_limit": 90,
 "points": 3
}
```

PERSONALITY & BEHAVIORAL ASSESSMENTS

2.1 BIG FIVE PERSONALITY ASSESSMENT

Purpose: Understand candidate's personality traits, work style preferences, and behavioral tendencies.

Objective: Assess cultural fit, team compatibility, and leadership potential using validated personality frameworks.

Test Structure

- **Duration:** 20 minutes
- **Questions:** 80 questions (16 per factor)
- **Framework:** Big Five Personality Model (OCEAN)
- **Openness:** Creativity, intellectual curiosity (16 questions)
- **Conscientiousness:** Organization, responsibility (16 questions)
- **Extraversion:** Social energy, assertiveness (16 questions)
- **Agreeableness:** Cooperation, empathy (16 questions)
- **Neuroticism:** Emotional stability, stress tolerance (16 questions)

Question Categories **Openness to Experience:**

```
{
 "question_id": "PER_001",
 "factor": "openness",
 "question": "I enjoy exploring new ideas and concepts.",
 "response_type": "likert_5",
 "scale": [
 {"value": 1, "label": "Strongly Disagree"},
 {"value": 2, "label": "Disagree"},
 {"value": 3, "label": "Neutral"},
 {"value": 4, "label": "Agree"},
 {"value": 5, "label": "Strongly Agree"}
 ],
 "keying": "positive",
 "facet": "intellectual_curiosity"
}
```

Conscientiousness:

```
{
 "question_id": "PER_017",
 "factor": "conscientiousness",
 "question": "I often leave tasks unfinished.",
 "response_type": "likert_5",
 "keying": "negative",
 "facet": "self_discipline"
}
```

Extraversion:

```
{
 "question_id": "PER_033",
 "factor": "extraversion",
 "question": "I feel energized when working with a team.",
 "response_type": "likert_5",
 "keying": "positive",
 "facet": "social_energy"
}
```

Agreeableness:

```
{
 "question_id": "PER_049",
 "factor": "agreeableness",
 "question": "I try to understand different perspectives before making judgments.",
 "response_type": "likert_5",
 "keying": "positive",
 "facet": "empathy"
}
```

Neuroticism (Emotional Stability):

```
{
 "question_id": "PER_065",
 "factor": "neuroticism",
 "question": "I remain calm under pressure.",
 "response_type": "likert_5",
 "keying": "negative",
 "facet": "stress_tolerance"
}
```

Scoring Algorithm

```
def calculate_personality_scores(responses):
 factor_scores = {
 'openness': 0,
 'conscientiousness': 0,
 'extraversion': 0,
 'agreeableness': 0,
 'neuroticism': 0
 }

 for response in responses:
 question = get_question(response.question_id)
 score = response.value

 # Reverse score for negative keying
 if question.keying == 'negative':
 score = 6 - score

 factor_scores[question.factor] += score

 # Convert to percentiles (0-100)
 percentile_scores = {}
 for factor, raw_score in factor_scores.items():
 percentile_scores[factor] = {
 'raw_score': raw_score,
 'percentile': calculate_percentile(raw_score, factor),
 'description': get_personality_description(factor, raw_score)
 }

 return {
 'factor_scores': percentile_scores,
 'role_compatibility': calculate_role_compatibility(percentile_scores),
 'team_dynamics': predict_team_fit(percentile_scores)
 }
```

Role Compatibility Matching

```
def calculate_role_compatibility(personality_scores):
 role_profiles = {
 'leadership_roles': {
 'extraversion': 0.8, 'conscientiousness': 0.9,
 'openness': 0.7, 'agreeableness': 0.6, 'neuroticism': 0.2
 },
 'analytical_roles': {
 'conscientiousness': 0.9, 'openness': 0.8,
 'extraversion': 0.4, 'agreeableness': 0.5, 'neuroticism': 0.3
 },
 'creative_roles': {
 'openness': 0.9, 'extraversion': 0.6,
 'conscientiousness': 0.6, 'agreeableness': 0.7, 'neuroticism': 0.4
 },
 'team_collaboration': {
 'agreeableness': 0.8, 'extraversion': 0.7,
 'conscientiousness': 0.7, 'openness': 0.6, 'neuroticism': 0.3
 }
 }

 compatibility_scores = {}
 for role, profile in role_profiles.items():
 score = calculate_profile_match(personality_scores, profile)
 compatibility_scores[role] = score

 return compatibility_scores
```

2.2 SITUATIONAL JUDGMENT TEST (SJT)

Purpose: Evaluate how candidates handle workplace scenarios and make decisions under pressure.

Objective: Assess practical judgment, decision-making skills, and behavioral competencies in work contexts.

Test Structure

- **Duration:** 25 minutes
- **Scenarios:** 15 workplace situations
- **Competencies Tested:**
- Leadership (3 scenarios)
- Communication (3 scenarios)
- Problem-solving (3 scenarios)
- Ethics (3 scenarios)
- Teamwork (3 scenarios)

Sample Scenarios Leadership Scenario:

```
{
 "scenario_id": "SJT_001",
 "competency": "leadership",
 "title": "Team Performance Management",
 "description": "You are leading a project team of 5 members. One team member consistently misses deadlines
 "responses": [
 {
 "id": "A",
 "text": "Speak directly with the underperforming team member about their contributions and set clear expec
 "competencies": ["leadership", "communication", "problem_solving"],
 "effectiveness_score": 4
 },
 {
 "id": "B",
 "text": "Report the situation to your manager immediately and request guidance",
 "competencies": ["escalation", "transparency"],
 "effectiveness_score": 2
 },
 {
```

```
 "id": "C",
 "text": "Redistribute the workload among other team members to ensure deadline compliance",
 "competencies": ["problem_solving", "time_management"],
 "effectiveness_score": 2
 },
 {
 "id": "D",
 "text": "Document all instances of poor performance and continue monitoring without intervention",
 "competencies": ["documentation"],
 "effectiveness_score": 1
 }
 ],
 "evaluation": {
 "most_effective": "A",
 "least_effective": "D",
 "reasoning": "Direct communication addresses the root cause and provides opportunity for improvement, whil
 }
}
```

Ethics Scenario:

```
{
 "scenario_id": "SJT_008",
 "competency": "ethics",
 "title": "Confidential Information Handling",
 "description": "During a client meeting, you accidentally overhear a confidential conversation about your
 "responses": [
 {
 "id": "A",
 "text": "Use the information to adjust your project strategy without mentioning the source",
 "competencies": ["strategic_thinking"],
 "effectiveness_score": 1
 },
 {
 "id": "B",
 "text": "Ignore the information completely and continue with your original plan",
 "competencies": ["ethics", "integrity"],
 "effectiveness_score": 4
 },
 {
 "id": "C",
 "text": "Report the incident to your manager and ask for guidance on how to proceed",
 "competencies": ["ethics", "transparency", "communication"],
 "effectiveness_score": 4
 },
 {
 "id": "D",
 "text": "Share the information with your team but advise them to keep it confidential",
 "competencies": ["team_communication"],
 "effectiveness_score": 1
 }
 ],
 "evaluation": {
 "most_effective": "B or C",
 "least_effective": "A",
 "reasoning": "Ethical behavior requires either ignoring unethically obtained information or seeking proper
 }
}
```

Scoring Algorithm

```
def calculate_sjt_scores(responses):
 competency_scores = {
 'leadership': 0,
 'communication': 0,
 'problem_solving': 0,
 'ethics': 0,
 'teamwork': 0
 }

 for response in responses:
 scenario = get_scenario(response.scenario_id)
 selected_response = scenario.responses[response.selected_option]

 # Add effectiveness score
 competency_scores[scenario.competency] += selected_response.effectiveness_score

 # Bonus for identifying most/least effective options correctly
 if response.most_effective == scenario.evaluation.most_effective:
 competency_scores[scenario.competency] += 1
 if response.least_effective == scenario.evaluation.least_effective:
 competency_scores[scenario.competency] += 1

 # Normalize scores and calculate percentiles
 final_scores = {}
 for competency, score in competency_scores.items():
 max_possible = get_max_score(competency)
 percentage = (score / max_possible) * 100
 final_scores[competency] = {
 'raw_score': score,
 'percentage': percentage,
 'percentile': calculate_percentile(percentage, f'sjt_{competency}'),
 'description': get_competency_description(competency, percentage)
 }

 return final_scores
```

TECHNICAL SKILLS ASSESSMENTS

3.1 SOFTWARE ENGINEERING ASSESSMENT

Purpose: Validate programming skills, algorithmic thinking, and software development capabilities.

Objective: Assess coding proficiency, problem-solving approach, and technical knowledge relevant to software engineering roles.

Test Structure

- **Duration:** 90 minutes
- **Components:**
- Algorithm problems (60 minutes - 4 problems)
- Code review questions (15 minutes - 2 questions)
- System design basics (15 minutes - 1 question)

Algorithm Problems by Difficulty **Easy Level (Beginner - 1 problem):**

```
{
 "problem_id": "ALG_001",
 "difficulty": "easy",
 "title": "Two Sum",
 "description": "Given an array of integers nums and an integer target, return indices of the two numbers s
 "example": {
 "input": "nums = [2,7,11,15], target = 9",
 "output": "[0,1]",
 "explanation": "Because nums[0] + nums[1] == 9, we return [0, 1]."
 },
 "constraints": [
 "2 <= nums.length <= 10^4",
 "-10^9 <= nums[i] <= 10^9",
 "-10^9 <= target <= 10^9",
 "Only one valid answer exists"
 ],
 "starter_code": {
 "python": "def twoSum(nums, target):\n # Your code here\n pass",
 "javascript": "function twoSum(nums, target) {\n // Your code here\n}",
 "java": "public int[] twoSum(int[] nums, int target) {\n // Your code here\n}"
 },
 "test_cases": [
 {"input": {"nums": [2,7,11,15], "target": 9}, "expected": [0,1]},
 {"input": {"nums": [3,2,4], "target": 6}, "expected": [1,2]},
 {"input": {"nums": [3,3], "target": 6}, "expected": [0,1]}
 ],
 "time_limit": 15,
 "points": 20
}
```

Medium Level (Intermediate - 2 problems):

```
{
 "problem_id": "ALG_015",
 "difficulty": "medium",
 "title": "Longest Palindromic Substring",
 "description": "Given a string s, return the longest palindromic substring in s.",
 "example": {
 "input": "s = 'babad'",
 "output": "'bab'",
 "explanation": "Note that 'aba' is also a valid answer."
 },
 "constraints": [
 "1 <= s.length <= 1000",
 "s consist of only digits and English letters"
 ],
 "starter_code": {
 "python": "def longestPalindrome(s):\n # Your code here\n pass"
 },
 "test_cases": [
 {"input": {"s": "babad"}, "expected": "bab"},
 {"input": {"s": "cbbd"}, "expected": "bb"},
 {"input": {"s": "a"}, "expected": "a"}
 ],
 "time_limit": 25,
 "points": 35
}
```

Hard Level (Advanced - 1 problem):

```
{
 "problem_id": "ALG_030",
 "difficulty": "hard",
 "title": "Median of Two Sorted Arrays",
 "description": "Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of
 "example": {
 "input": "nums1 = [1,3], nums2 = [2]",
 "output": "2.0",
 "explanation": "merged array = [1,2,3] and median is 2."
 },
 "constraints": [
 "nums1.length == m",
 "nums2.length == n",
 "0 <= m <= 1000",
 "0 <= n <= 1000",
 "1 <= m + n <= 2000"
 ],
 "time_limit": 45,
 "points": 50
}
```

Code Review Questions

```
{
 "question_id": "REV_001",
 "type": "code_review",
 "title": "Bug Identification",
 "code": "def binary_search(arr, target):\n left, right = 0, len(arr)\n while left < right:\n mid = (left +
 "question": "Identify the bug in this binary search implementation and explain how to fix it.",
 "expected_answer": "The bug is in the initialization: right should be len(arr) - 1, not len(arr). This cau
 "points": 15
}
```

Scoring Algorithm

```
def calculate_coding_score(submissions):
 total_score = 0
 category_scores = {
 'correctness': 0,
 'efficiency': 0,
 'code_quality': 0,
 'problem_solving': 0
 }

 for submission in submissions:
 problem = get_problem(submission.problem_id)

 # Test case results
 test_results = run_test_cases(submission.code, problem.test_cases)
 correctness = (test_results.passed / test_results.total) * problem.points

 # Time/space complexity analysis
 efficiency = analyze_complexity(submission.code, problem.expected_complexity)

 # Code quality metrics
 quality = analyze_code_quality(submission.code)
```

```
# Problem-solving approach
approach = evaluate_approach(submission.code, problem.optimal_approaches)

category_scores['correctness'] += correctness
category_scores['efficiency'] += efficiency
category_scores['code_quality'] += quality
category_scores['problem_solving'] += approach

total_score += correctness + efficiency + quality + approach

return {
'total_score': total_score,
'category_breakdown': category_scores,
'percentile': calculate_percentile(total_score, 'software_engineering'),
'skill_level': determine_skill_level(total_score),
'recommendations': generate_recommendations(category_scores)
}
```

3.2 MARKETING ASSESSMENT

Purpose: Evaluate strategic thinking, creative problem-solving, and marketing knowledge.

Objective: Assess marketing competencies, analytical skills, and ability to develop effective marketing strategies.

Test Structure

- **Duration:** 60 minutes
- **Components:**
- Case study analysis (35 minutes - 1 comprehensive case)
- Strategic questions (15 minutes - 5 questions)
- Creative brief (10 minutes - 1 exercise)

Comprehensive Case Study

```
{
 "case_id": "MKT_001",
 "title": "FinTech App Market Entry Strategy",
 "background": "EcoSpend is a new budgeting app targeting environmentally conscious millennials (ages 25-35
 "market_data": {
 "target_audience_size": "45 million US millennials",
 "current_budgeting_app_usage": "23% of target demographic",
 "eco_conscious_segment": "67% of millennials consider environmental impact in purchases",
 "average_customer_acquisition_cost": "$15-25 per user",
 "competition": [
 {"name": "Mint", "users": "20M", "strength": "comprehensive features"},
 {"name": "YNAB", "users": "4M", "strength": "budgeting methodology"},
 {"name": "PocketGuard", "users": "2M", "strength": "simplicity"}
 ]
 },
 "tasks": [
 {
 "task_id": "T1",
 "task": "Go-to-Market Strategy",
 "type": "strategic_planning",
 "prompt": "Develop a comprehensive go-to-market strategy. Include target audience segmentation, positionin
 "evaluation_criteria": [
 "Strategic thinking and market understanding",
 "Clear audience segmentation and targeting",
 "Differentiated positioning against competitors",
 "Compelling value proposition development",
 "Feasibility and execution considerations"
 ],
 "time_limit": 15,
 "points": 30
 },
 {
 "task_id": "T2",
 "task": "Budget Allocation",
 "type": "interactive_budget",
 "prompt": "Allocate the $100,000 budget across marketing channels to achieve 10,000 user acquisition goal.
 "channels": [
 {"name": "Social Media Advertising", "estimated_cac": "$12-18", "reach": "high"},
 {"name": "Google Ads", "estimated_cac": "$20-30", "reach": "medium"},
 {"name": "Influencer Marketing", "estimated_cac": "$8-15", "reach": "medium"},
 {"name": "Content Marketing", "estimated_cac": "$5-10", "reach": "low"},
 {"name": "App Store Optimization", "estimated_cac": "$3-8", "reach": "medium"},
 {"name": "PR/Media Outreach", "estimated_cac": "$10-20", "reach": "low"}
 ],
 "evaluation_criteria": [
 "Budget allocation rationale and math accuracy",
 "Channel selection based on target audience",
 "Understanding of CAC and ROI principles",
 "Risk diversification across channels",
 "Timeline and scaling considerations"
 ],
 "time_limit": 10,
 "points": 25
 },
 {
 "task_id": "T3",
 "task": "Campaign Development",
 "type": "creative_strategy",
 "prompt": "Create a campaign concept for the app launch. Include campaign theme, key messages, and channel
 "evaluation_criteria": [
 "Creative concept development and originality",
 "Alignment with brand positioning and audience",
 "Cross-channel integration and consistency",
 "Clear call-to-action and conversion strategy",
 "Measurable objectives and success metrics"
 ],
 "time_limit": 10,
 "points": 20
 }
 ]
}
```

Strategic Knowledge Questions

```
{
 "question_id": "MKT_Q001",
 "category": "digital_marketing",
 "question": "What is the primary advantage of using lookalike audiences in Facebook advertising?",
 "options": [
 "Lower cost per click compared to interest-based targeting",
 "Reaching users similar to your best existing customers",
 "Guaranteed higher conversion rates",
 "Access to more detailed demographic data"
 ],
 "correct_answer": "Reaching users similar to your best existing customers",
 "explanation": "Lookalike audiences use Facebook's algorithm to find users who share characteristics with
 "points": 5
}
```

Creative Brief Exercise

```
{
 "exercise_id": "MKT_BRIEF_001",
 "title": "Social Media Campaign Brief",
 "scenario": "Create a social media campaign brief for EcoSpend's Instagram launch targeting eco-conscious
 "requirements": [
 "Campaign objective and goals",
 "Target audience description",
 "Key messaging and tone",
 "Content types and posting strategy",
 "Success metrics and KPIs"
 ],
 "evaluation_rubric": {
 "strategic_alignment": 25,
 "audience_understanding": 20,
 "creative_execution": 20,
 "measurability": 20,
 "feasibility": 15
 },
 "time_limit": 10,
 "points": 25
}
```

Scoring Algorithm

```
def calculate_marketing_score(responses):
 category_scores = {
 'strategic_thinking': 0,
 'analytical_skills': 0,
 'creative_execution': 0,
 'market_knowledge': 0,
 'campaign_development': 0
 }

 # Case study evaluation
 case_responses = get_case_responses(responses)
 for task_response in case_responses:
 score = evaluate_marketing_response(task_response)
 category_scores.update(score.category_breakdown)

 # Knowledge questions
 knowledge_score = evaluate_knowledge_questions(responses.knowledge_questions)
 category_scores['market_knowledge'] += knowledge_score

 # Creative brief
```

```
brief_score = evaluate_creative_brief(responses.creative_brief)
category_scores['creative_execution'] += brief_score.creativity
category_scores['strategic_thinking'] += brief_score.strategy

total_score = sum(category_scores.values())

return {
'total_score': total_score,
'category_breakdown': category_scores,
'percentile': calculate_percentile(total_score, 'marketing'),
'competency_level': determine_marketing_level(total_score),
'strengths': identify_strengths(category_scores),
'development_areas': identify_weaknesses(category_scores)
}
```

ASSESSMENT IMPLEMENTATION REQUIREMENTS

Technology Stack

- **Frontend:** React with interactive components for visual questions
- **Backend:** Django with assessment engine and scoring algorithms
- **Database:** PostgreSQL with optimized schemas for questions and results
- **Media:** SVG generation for visual patterns, chart rendering for data interpretation
- **Security:** Sandboxed code execution for programming assessments

Quality Assurance Standards

- **Reliability:** Test-retest correlation >0.85 for all assessments
- **Validity:** Content validation by subject matter experts
- **Fairness:** Bias testing across demographic groups
- **Accessibility:** WCAG 2.1 AA compliance for all interfaces

Performance Requirements

- **Load Time:** <2 seconds for assessment initialization
- **Response Time:** <500ms for question navigation
- **Concurrent Users:** Support 1000+ simultaneous assessments
- **Data Integrity:** Real-time backup and recovery systems

This comprehensive guide provides all the detailed specifications needed to implement a world-class skills validation platform. Each assessment type includes sample questions, scoring algorithms, and technical requirements for full implementation.