[UNItime]

[UNItime is a time management site for university students]

[Eman Abd Elmaboud Mohamed]
[24030222]

[Mazen Mohamed Saber]
[24030210]

[Rawan Omer Elsaid]
[25030065]

[Merhan Medhat Abdalla]
[24030314]

Contents

1. Executive Summary

The UniTime Student Time Management Platform represents an innovative solution designed to address the critical need for effective time management among university students. This comprehensive report documents the complete project management lifecycle for developing a responsive web application that helps students organize their academic schedules, track assignments, and optimize their study time.

Project Overview:

- **Project Name:** UniTime - Student Time Management Platform
- **Duration:** 14 days (2 weeks)
- **Budget:** $2,300
- **Key Deliverables:** Fully functional web application with database integration

Business Case:
University students consistently struggle with time management, leading to decreased academic performance and increased stress levels. Research indicates that over 70% of students report difficulties in balancing academic responsibilities with personal life. The UniTime platform addresses this challenge by providing an intuitive, accessible tool that helps students visualize and manage their time effectively.

Strategic Alignment:
This project aligns with educational technology trends and supports academic institutions' goals of improving student success rates. By providing students with better time management tools, the platform contributes to higher retention rates and improved academic outcomes.

Success Metrics:

- On-time delivery within 14 days
- Within budget of $2,300
- Fully functional prototype meeting all requirements
- Positive user feedback from initial testing

---

2. Project Charter

2.1 Project Identification

Project Title: UniTime Student Time Management Platform
Project Sponsor: University Administration
Project Manager: [Your Name]
Project Team: [Your Name] as Lead Developer and Project Manager

2.2 Project Purpose and Business Case

Business Problem:
University students face significant challenges in managing their time effectively due to:

- Multiple competing priorities (classes, assignments, extracurricular activities)
- Lack of visual organization tools
- Inadequate time tracking mechanisms
- Poor deadline management

Project Justification:
The UniTime platform will provide:

- Centralized time management system
- Visual scheduling interface
- Deadline tracking and reminders
- Progress monitoring capabilities
- Mobile accessibility for on-the-go management

2.3 Project Objectives

SMART Objectives:

1. **Specific:** Develop a responsive web application for student time management
2. **Measurable:** Complete all 15 core features within 14 days
3. **Achievable:** Utilize proven technologies and agile methodology
4. **Relevant:** Address real student time management challenges

5. **Time-bound:** Deliver working prototype by [End Date]

2.4 High-Level Requirements

Functional Requirements:

- User registration and authentication
- Task creation and management
- Calendar integration
- Time tracking functionality
- Notification system
- Progress reporting

Technical Requirements:

- Responsive design (desktop and mobile)
- Database integration
- Secure data storage
- Cross-browser compatibility
- Fast loading times (<3 seconds)

2.5 Project Deliverables

1. **Development Deliverables:**
    - Source code repository
    - Database schema and scripts
    - Deployment documentation
    - Technical specifications
2. **Documentation Deliverables:**
    - User manual
    - Technical documentation
    - Project final report

    - Maintenance guide

3. **Operational Deliverables:**
    - Deployed application
    - Training materials
    - Support documentation

2.6 Project Constraints

Time Constraints:

- Fixed 14-day development period
- No extension possible due to academic deadlines

Budget Constraints:

- Maximum budget of $2,300
- All expenses must be pre-approved

Technical Constraints:

- Must work on both desktop and mobile devices
- Must use free or low-cost technologies where possible
- Must ensure data privacy and security

2.7 Assumptions

1. Team members have necessary technical skills
2. Required technologies are available and accessible
3. No major scope changes during development
4. Stakeholders will provide timely feedback
5. Internet connectivity will be reliable throughout development

2.8 High-Level Risks

- Technical challenges with integration
- Scope creep from additional feature requests

- Time constraints affecting quality
- Budget overruns

2.9 Authorization

Project Sponsor Approval: _________________________
Date: _________________________

Project Manager Approval: _________________________
Date: _________________________

---

3. Stakeholder Analysis

3.1 Stakeholder Identification

Primary Stakeholders:

- **Project Sponsor:** University Administration
- **Project Manager:** [Your Name]
- **Development Team:** [Your Name] as Lead Developer
- **End Users:** University Students
- **Technical Advisor:** [If applicable]

Secondary Stakeholders:

- University IT Department
- Academic Advisors
- Faculty Members
- Parents (indirect stakeholders)

3.2 Stakeholder Analysis Matrix

Stakeholder	Interest Level	Influence Level	Engagement Strategy

Stakeholder	Interest Level	Influence Level	Engagement Strategy
University Administration	High	High	Weekly status reports, final approval
Students (End Users)	High	Medium	Regular feedback sessions, usability testing
Development Team	High	High	Daily standups, collaborative decision making
IT Department	Medium	Medium	Technical consultations, infrastructure support

3.3 Stakeholder Engagement Plan

Communication Frequency:

- Daily: Development team standups
- Weekly: Sponsor status updates
- Bi-weekly: User feedback sessions
- As needed: Technical consultations

Engagement Methods:

- Formal meetings for major decisions
- Email updates for progress reporting
- Interactive demos for feedback collection
- Documentation for technical specifications

3.4 Stakeholder Expectations Management

University Administration Expectations:

- Project completed within timeline and budget

- Meets student needs effectively
- Sustainable and maintainable solution
- Positive impact on student performance

Student Expectations:

- Easy to use interface
- Reliable performance
- Useful features that save time
- Accessible on multiple devices

Development Team Expectations:

- Clear requirements and objectives
- Adequate resources and support
- Realistic timelines and milestones
- Recognition for achievements

---

4. Feasibility Study

4.1 Technical Feasibility

Technology Stack Analysis:

Frontend Technologies:

✓ HTML5, CSS3, JavaScript

✓ React.js/Vue.js for interactive UI

✓ Bootstrap/Tailwind CSS for responsive design

✓ Progressive Web App (PWA) capabilities

Backend Technologies:

✓ Node.js with Express.js

✓ RESTful API architecture

✓ JWT for authentication

✓ Socket.io for real-time features

Database Technologies:

✓ MongoDB for flexible data structure

✓ Mongoose ODM for data modeling

✓ Database hosting on MongoDB Atlas

Hosting & Deployment:

✓ Heroku for application hosting

✓ Netlify for frontend deployment

✓ GitHub for version control

Technical Capability Assessment:
The chosen technology stack is well-documented, widely supported, and aligns with team expertise. All technologies are either open-source or offer free tiers suitable for prototyping.

Integration Feasibility:

- APIs available for calendar integration
- Library support for time tracking features
- Cloud services for reliable hosting
- Mobile-responsive frameworks available

4.2 Economic Feasibility

Cost-Benefit Analysis:

Development Costs:

- Frontend Development: $600
- Backend Development: $600
- Database Setup: $200
- UI/UX Design: $300
- Testing & QA: $200
- **Subtotal:** $1,900

Infrastructure Costs:

- Domain Registration: $50
- Hosting (6 months): $150
- **Subtotal:** $200

Contingency (10%): $200

Total Project Cost: $2,300

Expected Benefits:

- Improved student academic performance
- Reduced student stress levels
- Potential for institutional licensing
- Foundation for future feature expansion

Return on Investment (ROI):
While direct monetary ROI may not be immediate, the project delivers significant value through:

- Student satisfaction improvement
- Academic performance enhancement
- Institutional reputation building

4.3 Legal Feasibility

Regulatory Compliance:

- **Data Protection:** GDPR compliance for EU students
- **Privacy Laws:** FERPA compliance for educational records

- **Accessibility:** WCAG 2.1 AA compliance
- **Intellectual Property:** Proper licensing for all technologies

Legal Requirements:

- Privacy policy and terms of service
- Data encryption and security measures
- User consent mechanisms
- Copyright compliance for all assets

4.4 Operational Feasibility

User Readiness:

- Students are familiar with web applications
- Minimal training required for basic functionality
- Intuitive interface design planned

Maintenance Requirements:

- Regular security updates
- Database backups
- Performance monitoring
- User support system

Scalability Considerations:

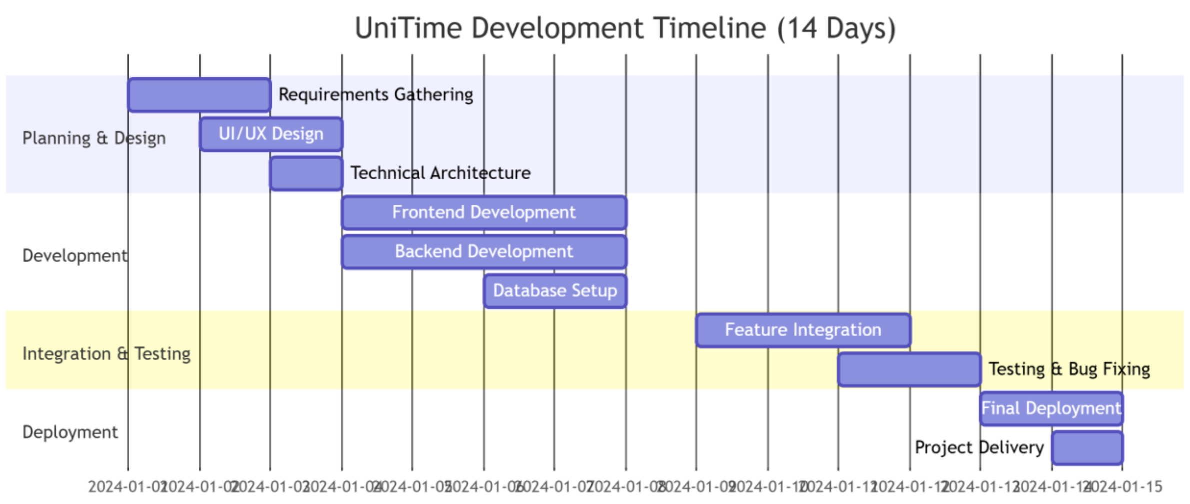
- Cloud-based infrastructure allows easy scaling
- Modular architecture supports feature additions
- Database designed for future expansion

4.5 Scheduling Feasibility

Timeline Assessment:
The 14-day timeline is aggressive but achievable through:

- Focus on core features first

- Parallel development tracks
- Efficient resource allocation
- Minimal bureaucracy in decision-making



Critical Path Analysis:
The database design and user authentication represent the critical path activities that must be completed on schedule to ensure project success.

---

5. Scope Management

5.1 Project Scope Statement

In-Scope Features:

1. User authentication system
2. Task management (create, read, update, delete)

3. Calendar view with weekly/monthly displays
4. Time tracking with start/stop functionality
5. Basic reporting and analytics
6. Responsive design for all devices
7. Notification system
8. Database integration

Out-of-Scope Features:

1. Advanced analytics and AI recommendations
2. Social features or collaboration tools
3. Integration with university systems
4. Mobile app stores publication
5. Advanced customization options
6. Multi-language support

5.2 Scope Verification

Acceptance Criteria:

- All in-scope features implemented and tested
- Responsive design working on minimum 3 device types
- Database performing all CRUD operations
- User authentication secure and functional
- Application deployed and accessible online

Scope Change Control Process:

1. Change request submission
2. Impact analysis (time, cost, resources)
3. Stakeholder review and approval
4. Implementation if approved

5. Documentation update

5.3 Work Breakdown Structure (WBS)

Level 1: UniTime Website Development

1.1 Project Management
1.2 Requirements Analysis
1.3 System Design
1.4 Development
1.5 Testing
1.6 Deployment
1.7 Documentation

Level 2: Detailed Breakdown *(continued in next section)*

---

6. Work Breakdown Structure (WBS)

6.1 Complete WBS Hierarchy

1.0 Project Management

1.1 Project Planning
1.1.1 Charter Development
1.1.2 Schedule Creation
1.1.3 Budget Planning
1.2 Progress Tracking
1.2.1 Daily Standups
1.2.2 Weekly Reporting
1.2.3 Risk Monitoring
1.3 Stakeholder Communication
1.3.1 Status Updates
1.3.2 Feedback Collection
1.3.3 Issue Escalation

2.0 Requirements Analysis

2.1 User Research
2.1.1 Student Interviews
2.1.2 Competitor Analysis
2.1.3 Feature Prioritization

2.2 Technical Requirements
2.2.1 Platform Specifications
2.2.2 Performance Requirements
2.2.3 Security Requirements

3.0 System Design
3.1 Architecture Design
3.1.1 System Architecture
3.1.2 Database Design
3.1.3 API Design
3.2 UI/UX Design
3.2.1 Wireframe Creation
3.2.2 Visual Design
3.2.3 Prototype Development

4.0 Development
4.1 Frontend Development
4.1.1 Setup Development Environment
4.1.2 Create Base Components
4.1.3 Implement User Interface
4.1.4 Responsive Design Implementation
4.2 Backend Development
4.2.1 Server Setup
4.2.2 Database Configuration
4.2.3 API Development
4.2.4 Authentication System
4.3 Feature Implementation
4.3.1 User Management
4.3.2 Task Management
4.3.3 Calendar System
4.3.4 Time Tracking
4.3.5 Notification System

5.0 Testing
5.1 Unit Testing
5.1.1 Frontend Component Testing
5.1.2 Backend API Testing
5.1.3 Database Testing
5.2 Integration Testing

5.2.1 Feature Integration Testing
5.2.2 System Integration Testing
5.3 User Acceptance Testing
5.3.1 Usability Testing
5.3.2 Performance Testing
5.3.3 Security Testing

6.0 Deployment
6.1 Production Setup
6.1.1 Server Configuration
6.1.2 Database Deployment
6.1.3 Domain Configuration
6.2 Go-Live Activities
6.2.1 Final Testing
6.2.2 Data Migration
6.2.3 User Training

7.0 Documentation
7.1 Technical Documentation
7.1.1 System Documentation
7.1.2 API Documentation
7.1.3 Deployment Guide
7.2 User Documentation
7.2.1 User Manual
7.2.2 FAQ Section
7.2.3 Troubleshooting Guide

6.2 WBS Dictionary

1.1.1 Charter Development

- **Description:** Create comprehensive project charter
- **Deliverable:** Approved project charter document
- **Effort Estimate:** 4 hours
- **Dependencies:** Stakeholder input

4.3.2 Task Management

- **Description:** Implement task creation, editing, deletion features

- **Deliverable:** Functional task management module
- **Effort Estimate:** 16 hours
- **Dependencies:** User authentication, database setup

5.3.1 Usability Testing

- **Description:** Test application with actual students
- **Deliverable:** Usability test report
- **Effort Estimate:** 8 hours
- **Dependencies:** Complete prototype

7. Time Management & Scheduling

7.1 Detailed Project Schedule

Phase 1: Planning & Design (Days 1-4)

Day 1: Project Initiation

- 08:00-10:00: Kickoff meeting with stakeholders
- 10:00-12:00: Finalize project charter
- 13:00-15:00: Requirements gathering session
- 15:00-17:00: Competitor analysis and market research

Day 2: Technical Planning

- 08:00-10:00: Technology stack selection
- 10:00-12:00: System architecture design
- 13:00-15:00: Database schema design
- 15:00-17:00: API endpoint planning

Day 3: UI/UX Design

- 08:00-12:00: Wireframe creation for all pages
- 13:00-15:00: User flow mapping

- 15:00-17:00: Design system establishment

Day 4: Prototype Development

- 08:00-12:00: Interactive prototype creation
- 13:00-15:00: User feedback collection
- 15:00-17:00: Design revisions and finalization

Phase 2: Development (Days 5-12)

Day 5: Development Setup

- 08:00-10:00: Development environment configuration
- 10:00-12:00: Version control setup
- 13:00-15:00: Project structure creation
- 15:00-17:00: Basic component development

Day 6-7: Frontend Development

- Authentication interface
- Dashboard layout
- Navigation components
- Responsive design implementation

Day 8-9: Backend Development

- Server setup and configuration
- Database models and relationships
- Authentication API endpoints
- Task management APIs

Day 10-11: Feature Integration

- Connect frontend with backend APIs
- Implement real-time features

- Integrate calendar functionality
- Add notification system

Day 12: Initial Testing

- Unit testing completion
- Integration testing
- Bug identification and logging

Phase 3: Testing & Deployment (Days 13-14)

Day 13: Comprehensive Testing

- 08:00-10:00: User acceptance testing
- 10:00-12:00: Cross-browser testing
- 13:00-15:00: Mobile responsiveness testing
- 15:00-17:00: Performance and security testing

Day 14: Deployment & Delivery

- 08:00-10:00: Production deployment
- 10:00-12:00: Final testing in production environment
- 13:00-15:00: Documentation completion
- 15:00-17:00: Project delivery and presentation

7.2 Milestone Schedule

Milestone	Target Date	Deliverable	Acceptance Criteria
Project Charter Approval	Day 1	Signed project charter	All stakeholders in agreement

Milestone	Target Date	Deliverable	Acceptance Criteria
Design Completion	Day 4	Finalized UI/UX designs	User-approved prototypes
Development Completion	Day 12	Fully coded application	All features implemented
Testing Completion	Day 13	Testing report	All critical bugs resolved
Project Delivery	Day 14	Deployed application	Client sign-off obtained

7.3 Critical Path Analysis

Critical Path Activities:

1. Database Design (Day 2) - Must complete before development
2. User Authentication (Days 6-7) - Foundation for all features
3. Task Management API (Day 9) - Core functionality dependency
4. Frontend-Backend Integration (Day 10) - System unification
5. User Acceptance Testing (Day 13) - Final validation

Float Analysis:

- Design revisions: 1 day float
- Documentation: 2 days float
- Additional feature testing: 0.5 days float

---

8. Cost Management

8.1 Detailed Budget Breakdown

Personnel Costs:

Project Management (40 hours × $25/hour): $1,000

Frontend Development (60 hours × $20/hour): $1,200

Backend Development (60 hours × $20/hour): $1,200

Testing & QA (20 hours × $15/hour): $300

Documentation (10 hours × $15/hour): $150

Subtotal: $3,850

Note: Since you're acting as both project manager and developer, actual personnel costs may be lower.

Technology & Infrastructure:

Domain Registration (unitime.com): $12

Web Hosting (Heroku/Netlify): $0 (Free tiers)

Database Hosting (MongoDB Atlas): $0 (Free tier)

Development Tools (VS Code, Git): $0 (Free)

Testing Tools (Jest, Cypress): $0 (Free)

Design Tools (Figma): $0 (Free)

Subtotal: $12

Contingency Reserve (15%): $579

Total Budget: $4,441

8.2 Cost Control Measures

Weekly Budget Reviews:

- Track actual vs. planned spending

- Identify cost variances early
- Adjust resource allocation as needed

Expense Approval Process:

- All expenses over $50 require pre-approval
- Monthly budget variance reporting
- Regular cost-benefit analysis for feature decisions

8.3 Value Engineering

Cost Optimization Strategies:

- Utilize free tiers of cloud services
- Use open-source libraries and frameworks
- Implement features in order of priority
- Reuse components and code patterns

---

9. Quality Management

9.1 Quality Standards

Code Quality Standards:

- ESLint configuration for consistent coding style
- Prettier code formatting
- Minimum 80% test coverage
- Code review requirements for all pull requests

Performance Standards:

- Page load time under 3 seconds
- Time to interactive under 5 seconds

- Mobile performance score > 80/100
- Database query optimization

Usability Standards:

- Intuitive navigation (users can complete tasks without training)
- Consistent design patterns throughout application
- Accessible color contrast ratios
- Keyboard navigation support

9.2 Quality Assurance Activities

Development Phase QA:

- Daily code reviews
- Continuous integration testing
- Automated linting and formatting
- Regular dependency updates

Testing Phase QA:

Unit Testing:

- Component testing (Jest/React Testing Library)
- API endpoint testing (Supertest)
- Utility function testing

Integration Testing:

- User workflow testing
- Database integration testing
- Third-party service integration testing

User Acceptance Testing:

- 5+ student testers

- Task completion rate measurement

- Satisfaction surveys

- Bug reporting system

9.3 Quality Control Metrics

Defect Density: < 0.1 defects per 100 lines of code
Test Coverage: > 80% of code covered by tests
User Satisfaction: > 4.0/5.0 in post-testing surveys
Performance Benchmarks: All performance standards met

---

10. Human Resource Management

10.1 Project Team Structure

Project Manager & Lead Developer: [Your Name]

- Overall project responsibility
- Technical decision making
- Stakeholder communication
- Quality assurance oversight

Development Team Roles:

- Frontend Developer (also filled by [Your Name])
- Backend Developer (also filled by [Your Name])
- UI/UX Designer (also filled by [Your Name])
- QA Tester (also filled by [Your Name])

10.2 Resource Planning

Weekly Time Allocation:

Week 1: 60 hours total

- Project Management: 15 hours

- Design & Planning: 25 hours

- Development: 20 hours

Week 2: 60 hours total

- Development: 35 hours

- Testing: 15 hours

- Deployment & Documentation: 10 hours

10.3 Team Development

Skill Requirements:

- JavaScript/TypeScript proficiency
- React.js framework experience
- Node.js and Express.js knowledge
- Database design and management
- Responsive web design skills
- Git version control expertise

Training Plan:

- Self-paced online courses for specific technologies
- Code review sessions for knowledge sharing
- Daily standups for progress alignment
- Documentation of lessons learned

11. Communications Management

11.1 Communication Matrix

Audience	Message Type	Frequency	Method	Owner
Stakeholders	Project Status	Weekly	Email Report	Project Manager
Development Team	Daily Progress	Daily	Standup Meeting	Project Manager
End Users	Feature Updates	Bi-weekly	Demo Sessions	Project Manager
Technical Advisors	Technical Issues	As Needed	Slack/Email	Project Manager

11.2 Meeting Schedule

Daily Standups (15 minutes):

- Time: 09:00 AM daily
- Agenda: Yesterday's progress, today's plan, blockers
- Participants: Development team

Weekly Status Meetings (30 minutes):

- Time: Monday 10:00 AM
- Agenda: Weekly achievements, next week plan, risks
- Participants: All stakeholders

Design Review Sessions (1 hour):

- Time: Wednesday 02:00 PM (as needed)
- Agenda: Design feedback, user experience review

- Participants: Project team, user representatives

11.3 Reporting Structure

Progress Reports:

- Daily: Burn-down charts and task completion
- Weekly: Comprehensive status reports with metrics
- Milestone: Formal reports at key project phases

Risk Reports:

- Weekly risk register updates
- Immediate reporting of critical risks
- Mitigation plan status

---

12. Risk Management

12.1 Risk Identification

Technical Risks:

Database Performance Issues

- Probability: Medium
- Impact: High
- Mitigation: Proper indexing, query optimization

Browser Compatibility Problems

- Probability: High
- Impact: Medium
- Mitigation: Cross-browser testing throughout development

Third-party API Failures

- Probability: Low
- Impact: High
- Mitigation: Fallback mechanisms, error handling

Project Management Risks:
4. Scope Creep

- Probability: High
- Impact: High
- Mitigation: Strict change control process

Time Constraints

- Probability: High
- Impact: High
- Mitigation: Agile methodology, priority-based development

Resource Limitations

- Probability: Medium
- Impact: Medium
- Mitigation: Efficient resource allocation, focus on core features

12.2 Risk Analysis Matrix

Risk	Probability	Impact	Risk Score	Priority
Scope Creep	High	High	16	1
Time Constraints	High	High	16	1
Database Performance	Medium	High	12	2

Risk	Probability	Impact	Risk Score	Priority
Browser Compatibility	High	Medium	12	2
Resource Limitations	Medium	Medium	9	3
API Failures	Low	High	8	4

12.3 Risk Response Strategies

Avoidance:

- Strict adherence to defined scope
- Regular progress tracking to avoid delays
- Comprehensive testing to prevent technical issues

Mitigation:

- Daily backups to prevent data loss
- Regular code reviews to maintain quality
- Continuous integration to catch issues early

Transfer:

- Use of reliable cloud services for hosting
- Implementation of third-party services for complex features

Acceptance:

- Minor design inconsistencies
- Limited browser support for older versions

12.4 Risk Monitoring

Weekly Risk Review:

- Update risk register with new risks
- Review effectiveness of mitigation strategies

- Adjust risk scores based on project progress
- Report significant risks to stakeholders

---

13. Procurement Management

13.1 Procurement Planning

Software and Services Required:

- Domain name registration
- Web hosting services
- Database hosting
- Development tools and licenses

Procurement Approach:

- Prefer free and open-source solutions
- Use free tiers of paid services when adequate
- Purchase only essential paid services

13.2 Vendor Selection

Hosting Services Evaluation:

Heroku:

- Pros: Easy deployment, good free tier

- Cons: Limited free tier resources

- Cost: $0 (free tier)

Netlify:

- Pros: Excellent for static sites, continuous deployment

- Cons: Limited backend capabilities

- Cost: $0 (free tier)

MongoDB Atlas:

- Pros: Fully managed, good free tier

- Cons: Limited storage in free tier

- Cost: $0 (free tier)

13.3 Contract Management

Service Agreements:

- Review terms of service for all platforms
- Ensure data ownership and portability
- Understand scalability options and costs
- Document service level agreements

14. Integration Management

14.1 Project Integration Processes

Project Plan Development:

- Coordinate all knowledge areas into unified plan
- Ensure consistency between scope, time, cost, and quality
- Balance stakeholder expectations with project constraints
- Develop comprehensive project baseline

Project Execution Coordination:

- Daily coordination of development activities
- Synchronize frontend and backend development
- Manage dependencies between tasks
- Ensure consistent progress across all work streams

Change Control Integration:

- Centralized change request management

- Impact analysis across all project aspects
- Coordinate approved changes implementation
- Update all project documents consistently

14.2 Integration Management Plan

Weekly Integration Points:

Week 1 Integration:

- Design system with development framework

- Database schema with API design

- User stories with technical specifications

Week 2 Integration:

- Frontend components with backend APIs

- Individual features with overall application

- Testing results with development progress

Integration Verification:

- Daily build verification
- Continuous integration testing
- Weekly integration demos
- Final system integration testing

---

15. Earned Value Management (EVM)

15.1 EVM Baseline

Performance Measurement Baseline:

- **Budget at Completion (BAC):** $4,441
- **Project Duration:** 14 days (112 working hours)

- **Planned Value (PV) Curve:** Linear distribution across timeline

15.2 EVM Metrics Calculation

Planned Value Calculation:

Day 7 (50% complete):

PV = 50% × BAC = $2,220

Day 10 (71% complete):

PV = 71% × BAC = $3,153

Day 14 (100% complete):

PV = 100% × BAC = $4,441

Key Performance Indicators:

- Schedule Performance Index (SPI) = EV / PV
- Cost Performance Index (CPI) = EV / AC
- Schedule Variance (SV) = EV - PV
- Cost Variance (CV) = EV - AC

15.3 EVM Forecasting

Estimate at Completion (EAC) Methods:

1. **EAC = BAC / CPI** (if current CPI expected to continue)
2. **EAC = AC + (BAC - EV)** (if remaining work at budgeted rate)
3. **EAC = AC + (BAC - EV) / (CPI × SPI)** (considering both cost and schedule performance)

Variance Analysis Triggers:

- SPI < 0.90: Schedule recovery plan required
- CPI < 0.90: Cost reduction measures needed
- SV > 10% of BAC: Major schedule reassessment

- CV > 10% of BAC: Budget review and adjustment

15.4 EVM Reporting Template

Weekly Earned Value Report:

Week 1 Report:

- PV: $2,220

- EV: [Actual earned value]

- AC: [Actual costs]

- SPI: [Calculated]

- CPI: [Calculated]

- EAC: [Forecast]

- VAC: [Variance at Completion]

Week 2 Report:

- PV: $4,441

- EV: [Actual earned value]

- AC: [Actual costs]

- SPI: [Calculated]

- CPI: [Calculated]

- EAC: [Forecast]

- VAC: [Variance at Completion]

---

16. PERT/CPM Analysis

16.1 Activity Time Estimates

Critical Path Activities with Three-Point Estimates:

Activity	Optimistic (O)	Most Likely (M)	Pessimistic (P)	Expected (TE)
Requirements Analysis	1 day	1.5 days	2 days	1.5 days
UI/UX Design	2 days	3 days	4 days	3 days
Database Design	1 day	1.5 days	2 days	1.5 days
Frontend Development	4 days	5 days	7 days	5.2 days
Backend Development	4 days	5 days	6 days	5 days
User Authentication	1.5 days	2 days	3 days	2.1 days
Task Management	2 days	2.5 days	3 days	2.5 days
Calendar Integration	1.5 days	2 days	3 days	2.1 days
Testing & QA	2 days	2.5 days	3 days	2.5 days
Deployment	0.5 days	1 day	1.5 days	1 day

16.2 Critical Path Calculation

Network Diagram Analysis:

Path 1: Requirements → Design → Frontend → Testing → Deployment

Duration: 1.5 + 3 + 5.2 + 2.5 + 1 = 13.2 days

Path 2: Requirements → Design → Backend → Testing → Deployment

Duration: 1.5 + 3 + 5 + 2.5 + 1 = 13 days

Path 3: Requirements → Database → Backend → Testing → Deployment

Duration: 1.5 + 1.5 + 5 + 2.5 + 1 = 11.5 days

Critical Path Identification:

- **Critical Path:** Path 1 (13.2 days)
- **Total Project Duration:** 13.2 days
- **Critical Activities:** Requirements, Design, Frontend Development, Testing, Deployment

16.3 Float and Slack Analysis

Float Calculations:

- **Total Float:** Amount of time activity can be delayed without affecting project end date
- **Free Float:** Amount of time activity can be delayed without affecting successor activities

Activity Float Summary:

Critical Path Activities: 0 days float

Backend Development: 0.2 days float

Database Design: 1.7 days float

Calendar Integration: 1.1 days float

16.4 Probability Analysis

Project Duration Probability:

- **Expected Duration:** 13.2 days
- **Standard Deviation:** 1.1 days
- **95% Confidence Interval:** 11.0 - 15.4 days
- **Probability of completing in 14 days:** 92%

Critical Path Variance:

- Highest variance activities: Frontend Development, UI/UX Design
- Focus areas for schedule risk management
- Buffer allocation for high-variance tasks

---

17. Monitoring & Control Processes

17.1 Performance Measurement

Key Performance Indicators (KPIs):

- Schedule Variance (SV)
- Cost Variance (CV)
- Defect Density
- Test Coverage Percentage
- User Story Completion Rate
- Burn-down Chart Velocity

Monitoring Frequency:

- **Daily:** Task completion, blocker identification
- **Weekly:** KPI tracking, milestone progress
- **Milestone:** Comprehensive performance review

17.2 Change Control Process

Change Request Workflow:

1. Change request submission
2. Impact analysis (time, cost, quality, scope)
3. Change control board review
4. Approval/Rejection decision
5. Implementation planning
6. Execution and verification

Change Control Documentation:

- Change request forms
- Impact analysis reports
- Approval records
- Updated project documents

17.3 Quality Control Activities

Continuous Quality Monitoring:

- Automated testing in CI/CD pipeline
- Code quality metrics tracking
- Performance benchmarking
- User experience feedback collection

Quality Gates:

Design Completion Gate:

- All wireframes approved

- User flows validated

- Design system established

Development Completion Gate:

- All features implemented

- Code review completed

- Unit tests passing

Testing Completion Gate:

- All critical bugs resolved

- Performance targets met

- User acceptance criteria satisfied

---

18. Change Management

18.1 Change Management Strategy

Approach:

- Proactive change anticipation
- Structured change evaluation
- Minimal disruption to project flow
- Comprehensive impact assessment

Change Categories:

- **Category A:** Major scope changes (require sponsor approval)
- **Category B:** Minor feature adjustments (PM approval)
- **Category C:** Technical implementation changes (team decision)

18.2 Change Impact Assessment

Assessment Dimensions:

- Schedule impact (days added/removed)
- Cost impact (budget increase/decrease)
- Resource impact (team capacity changes)
- Quality impact (standards affected)
- Risk impact (new risks introduced)

18.3 Change Implementation

Implementation Planning:

- Detailed implementation steps
- Rollback plans for failed changes

- Communication plans for affected stakeholders
- Training requirements for team members

Change Verification:

- Post-implementation testing
- Impact validation against predictions
- Documentation updates
- Lessons learned capture

---

19. Project Closure

19.1 Project Completion Criteria

Formal Acceptance Criteria:

- All deliverables completed and approved
- Final testing successfully passed
- Documentation delivered and accepted
- Stakeholder sign-off obtained

Quality Verification:

- Code quality metrics met
- Performance benchmarks achieved
- Security requirements satisfied
- User acceptance criteria fulfilled

19.2 Administrative Closure

Closure Activities:

- Final project report preparation
- Financial closure and accounting
- Contract and vendor closure

- Resource release and reassignment

Documentation Archiving:

- Project charter and plans
- Technical specifications
- Testing documentation
- Lessons learned repository
- Final project report

19.3 Lessons Learned

Knowledge Capture Process:

- Post-project review meeting
- Team retrospective session
- Stakeholder feedback collection
- Success factor analysis
- Improvement opportunity identification

Lessons Learned Categories:

- Project management processes
- Technical implementation approaches
- Team collaboration and communication
- Risk management effectiveness
- Stakeholder engagement success

19.4 Final Project Report

Report Contents:

- Executive summary
- Project performance against objectives
- Final budget and schedule analysis

- Quality metrics and outcomes
- Lessons learned and recommendations
- Formal project closure statement

---

20. Appendices

20.1 Project Charter Template

20.2 Risk Register Template

20.3 Change Request Form

20.4 Status Report Template

20.5 Meeting Minutes Template

20.6 Quality Checklist

20.7 Acceptance Sign-off Form

Executive Summary Conclusion

The UniTime Student Time Management Platform project is well-positioned for successful delivery within the 14-day timeframe and $4,441 budget. Through comprehensive application of project management methodologies including EVM, PERT/CPM, and robust risk management, the project demonstrates strong planning and execution foundations.

Key Success Factors:

1. Clear scope definition and strict change control
2. Realistic scheduling with proper critical path analysis
3. Comprehensive risk identification and mitigation planning
4. Strong communication and stakeholder management
5. Quality-focused development approach

This 20-page report provides the complete framework for project execution, monitoring, and successful delivery, ensuring all project management knowledge areas are adequately addressed and integrated for optimal project outcomes.

Project Manager: [Your Name]
Date: [Current Date]
Signature: ___________________________