Software Project Management Plan (SPMP - Team1)

CS-673 (Software Engineering)

Fall 2024

By

Sergio Khalil

Yuhang Zhang

Asma Asiri

Brad Nissenbaum

Shivaang Kumar

Table of contents:

Project Organization

- **Process model:**

The project will follow an agile development methodology, incorporating iterative cycles of development, testing, and refinement based on feedback and requirements.

- **Organizational Structure:**

Project Manager: Brad Nissenbaum

Scrum Master: Sergio Khalil

Development Team:

- Backend Developers: Brad Nissenbaum, Sergio Khalil
- Frontend Developers: Yuhang Zhang, Asma Asiri, Shivaang Kumar

Quality Assurance (QA): Yuhang Zhang, Shivaang Kumar

UI/UX Designer: Asma Asiri

- **Project Responsibility:**
  - **Project Manager**: Oversees the project, manages risks, and communicates with stakeholders.
  - **Scrum Master**: Facilitates Scrum ceremonies, resolves blockers, and ensures adherence to Agile principles.
  - **Developers**: Responsible for the design, coding, and implementation of features.
  - **QA Engineers**: Test the application and ensure quality standards are met.
  - **UI/UX Designer**: Creates wireframes, and mockups, and ensures the product is user-friendly and visually appealing.

Managerial Process Plans

- **Start-Up Plan:**
  - **Initial Setup**: Setup repositories, configure project management tools (Jira), and establish communication channels (Slack).
  - **Kickoff Meeting**: Discuss project goals, and deliverables, and assign initial tasks.
- **Work Plan:**
  - **Task Breakdown**: Tasks are broken down into user stories and epics, and logged into Jira.
  - **Sprint Planning**: Sprints are planned bi-weekly, with defined goals and deliverables.
- **Control Plan:**
  - **Progress Monitoring**: Use Jira for tracking task completion and burndown charts.
  - **Change Management**: Evaluate and approve changes through Scrum meetings.
  - **Quality Control**: Regular code reviews and QA testing cycles.
- **Closeout Plan:**
  - **Project Closure**: Final sprint review, documentation handover, and stakeholder sign-off.
  - **Post-Release Evaluation**: Review the project's success and document lessons learned.

Project Management tools

1. **Jira**
   - Task management, sprint planning, and progress tracking.
   - Backlog management, Scrum, issue tracking, and reporting.
2. **GitHub**
   - Version control and collaborative development.
   - Repository management, branching strategy, and code reviews.
3. **Slack**
   - Team communication and collaboration.

    - Channels for focused discussions, direct messaging, and integration with Jira and GitHub for notifications.
4. **Google Drive**
    - Document storage and collaboration.
    - Document sharing, real-time editing, and version control.
5. **Figma**
    - UI/UX design and prototyping.
    - Design creation, interactive prototyping, and team collaboration on designs.

Technical Process Plans

- **Development Plan:**

Technology Stack:

- Frontend: React, JavaScript, HTML, CSS
- Backend: Node.js, MongoDB
- Real-time Updates: Socket.io

Development Phases:

- Phase 1: Requirements Gathering and Analysis
- Phase 2: UI/UX Design
- Phase 3: Backend and Frontend Development
- Phase 4: Integration and Testing
- Phase 5: Deployment and Monitoring

- **Methods, Tools, and Techniques:**

Development Tools: IDEs, Git, and continuous integration tools.

Testing Tools: Automated testing frameworks and manual testing protocols.

- **Infrastructure Plan:**

Development Environment: Local machines and cloud-based services.
Testing Environment: Separate environment with configurations matching the production setup.

Deployment Environment: Production environment with secure access controls.

Supporting Process Plans

- **Configuration Management Plan:**

Version Control: Feature branching strategy with pull requests reviewed before merging into the main branch.

- **Verification and Validation Plan:**

Unit Testing: Automated testing of individual components.
Integration Testing: Ensure modules work together as expected.
User Acceptance Testing: Stakeholder review and feedback.

- **Documentation Plan:**

Documentation Repository: Store all project documentation in Google Drive with version tracking.

- **Documentation Types:**
  - Requirement Documents
  - Design Specifications
  - User Manuals

- **Training Plan**

Team Training: Internal training sessions on new tools and techniques.
User Training: Create video tutorials and documentation for end-users.

Project Schedule

Sprint 1-2: Requirements and Design
Sprint 3-4: Initial Development and Backend Setup
Sprint 5-6: Frontend Development and Integration
Sprint 7-8: Testing and Quality Assurance
Sprint 9: Deployment and Final Review

Risk Management Plan

- **Risk Identification**
  - **Developmental Delays**: Potential for delays due to technical challenges.
  - **Scope Creep**: Uncontrolled changes in project scope.
  - **Team Member Unavailability**: Unexpected absence of key team members.
  - **Code Conflicts**: Merge conflicts due to parallel development.

- **Risk Mitigation**
  - **Regular Check-ins**: Weekly stand-ups to identify and resolve issues early.
  - **Scope Control**: Strict change management and scope review during Scrum meetings.
  - **Backup Resources**: Cross-training team members to cover for absences.
  - **Frequent Merges**: Regularly merging code to reduce conflict risks.

- **Risk Owners**
  - **Project Manager** is responsible for managing developmental delays and team member unavailability. They monitor the project schedule and resource allocation, facilitate risk identification, and implement resolutions during regular progress meetings. Additionally, they ensure comprehensive task allocation and documentation and maintain a backup plan to manage team resources effectively.

  - **Scrum Master** is accountable for controlling scope creep. They manage the project scope, handle change requests, and conduct regular stakeholder meetings to review and align scope and requirements. They ensure that all new requests are evaluated for their impact on the timeline and resources before approval.

  - **Backend Developer** oversees code merge conflicts. They are responsible for implementing the branching strategy and resolving conflicts during integration. They review and approve code merges to minimize conflicts and ensure code consistency across the project.

- **QA Engineer** is responsible for managing code quality issues. They establish coding standards and ensure adherence through regular code reviews. The QA Engineer also oversees testing processes, employing both automated and manual testing methods to maintain high code quality and early defect detection. They ensure the final product meets the required standards and is free of critical defects.

Resource Management Plan

- **Human Resources**
  - **Core Team**: Five members as described in the project organization.
  - **Additional Support**: External consultants as needed for specialized tasks.

- **Tools and Equipment**
  - **Development Tools**: Laptops, IDEs, and necessary software licenses.
  - **Collaboration Tools**: Slack for communication, Jira for project management.

Measurement and Reporting Plan

Progress Metrics: Track completed tasks, sprint velocity, and burn-down rates.
Quality Metrics: Defect density, code quality ratings, and test coverage.
Reporting Frequency: Weekly status reports to stakeholders.

Project Documentation and Monitoring

- **Documentation Plan**
  - **Project Requirements Document:**
    - **Description**: Captures all functional and non-functional requirements of the Real Estate Checklist Tracker project.
    - **Owner**: Project Manager
    - **Updates**: Updated at the end of Iteration 1 and whenever requirements change.
  - **Design Documentation:**
    - **Description**: Includes UI/UX designs, wireframes, and technical design documents.
    - **Owner**: UI/UX Designer

    - **Updates**: Updated after design reviews and stakeholder feedback during Iteration 1 and Iteration 2.
- **Technical Specifications:**
    - **Description**: Detailed documentation of the backend and frontend architecture, API specifications, and database schema.
    - **Owner**: Backend and Frontend Developers
    - **Updates**: Updated as part of each development iteration and after major changes.
- **User Documentation:**
    - **Description**: User manuals, tutorials, and guides for real estate agents, clients, and brokers.
    - **Owner**: QA Engineer
    - **Updates**: Finalized during the final testing phase in Iteration 4.
- **Deployment and Maintenance Guide:**
    - **Description**: Detailed steps for deploying the application to the production environment and maintaining it.
    - **Owner**: Backend Developer
    - **Updates**: Updated after each deployment and during final review in Iteration 5.
- **Project Reports:**
    - **Description**: Weekly status reports, sprint review documents, and final project reports.
    - **Owner**: Project Manager
    - **Updates**: Weekly updates throughout the project.

- **Monitoring Plan**
    - **Progress Reporting:**
        - **Weekly Status Reports**: Prepared by the Project Manager, these reports will include the progress of each task, any blockers, risk status, and upcoming milestones. Reports will be shared with all stakeholders.
        - **Sprint Review Meetings**: At the end of each sprint, a review meeting will be held to demonstrate completed work, discuss what went well, what could be improved, and plan for the next sprint.
        - **Daily Stand-Ups**: Short daily meetings where each team member provides updates on what they did yesterday, what they plan to do today, and any impediments.

- **Metrics and KPIs:**
    - **Velocity**: The number of story points completed in each sprint. This metric will help track the team's progress and forecast future sprint performance.
    - **Burndown Chart**: Tracks the remaining work in the sprint backlog. This visual tool will help the team see progress towards the sprint goal.
    - **Defect Density**: The number of defects found during testing per 1,000 lines of code. This will help monitor code quality over time.
    - **Lead Time**: The total time from when a task is created to completion. This will help identify bottlenecks in the workflow.
    - **Task Completion Rate**: The percentage of tasks completed on time compared to those planned. This will help assess team efficiency and identify any issues in task estimation.

Estimation costs

To estimate developer costs, we'll assume the following:

- **Backend Development**: 10,000 lines of code (LOC)
- **Frontend Development**: 8,000 lines of code (LOC)
- **Average Productivity**: 50 LOC/hour
- **Average Rate for Developers**: $50/hour

Backend Development Cost:

- 10,000 LOC ÷ 50 LOC/hour = 200 hours
- 200 hours × $50/hour = $10,000

Frontend Development Cost:

- 8,000 LOC ÷ 50 LOC/hour = 160 hours
- 160 hours × $50/hour = $8,000

Total Developer Cost:

- Backend: $10,000
- **Frontend:** $8,000
- **Total Human Resources Cost**: $18,000

2. Software Licenses & Tools:

- **Jira**: Free for teams of up to 10 users; assume free tier or institutional license.
- **GitHub**: Free for public repositories or included with educational licenses.
- **Slack**: Free version for basic communication.
- **Figma**: Free for individual users or included with educational licenses.

Total Software & Tools Cost: $0 (assuming institutional licenses or free versions)

Total Project Cost Estimate:

$18,000 (Human Resources) + $0 (Software Licenses & Tools) = **18000$**