Software Engineering Economics
Project Report
VacaNest

Aiman K21-3906
Huda K21-3907

Software Cost Estimation Techniques for VacaNest Development

1.Introduction

This report explores software cost estimation techniques for VacaNest, a vacation rental platform. By employing various methodologies, we aim to ensure accurate resource allocation and successful project management.

2.Methodologies

- LOC Calculation: Estimates lines of code.
- FPA: Assesses project functionality for holistic insights.
- COCOMO: Provides detailed resource requirements.
- COSYSMO: Considers risk and uncertainty for comprehensive estimation.
- Expert Judgment: Utilizes expert opinions for consensus-based estimates.
- Agile Techniques: Dynamic and collaborative estimation processes.

3.Conclusion

Effective cost estimation is crucial for VacaNest's success. By leveraging diverse techniques, stakeholders can anticipate project needs and facilitate informed decision-making.

Abstract:

This report presents a comprehensive exploration of software cost estimation techniques applied to the development of VacaNest, a hypothetical vacation rental/ property buying platform. Leveraging established methodologies such as Lines of Code (LOC) calculation, Function Point Analysis (FPA), COCOMO, and COSYSMO, alongside innovative approaches like Expert Judgment with Delphi Method and Agile Estimation Techniques, we aim to provide a thorough understanding of project estimation. Through the application of these methodologies, we seek to ensure accurate resource allocation, informed decision-making, and successful project management in a dynamic development environment.

LOC Calculation (Hypothetical)

1. **Backend (JavaScript):**
    - **Node.js:** Node.js will serve as the backend runtime environment, allowing JavaScript to run on the server-side. It will handle HTTP requests, business logic, and database operations.
        i. Estimated lines of code: Approximately 55,000 to 216,000 lines.

    - **Express.js** (or similar framework): Express.js is a popular web application framework for Node.js. It simplifies the process of building APIs and handling middleware. It will be used to define routes, handle requests, and manage server-side logic.
        i. Estimated lines of code: Approximately 15,000 to 52,000 lines.

    - **Database interactions** (e.g., using Sequelize or similar ORM): JavaScript code will be used to interact with the database, including defining models, executing queries, and handling data manipulation.
        i. Estimated lines of code: Approximately 5,500 to 22,000 lines.

2. **Frontend (React):**
    - **React:** React will continue to serve as the frontend framework for building the user interface and managing state. It will render components, handle user interactions, and communicate with the backend.
        i. Estimated lines of code: Approximately 33,000 to 150,000 lines.

3. **Database (Amazon RDS):**
    - **Database code:** JavaScript code (using Node.js) will handle database interactions, including defining schemas, executing queries, and managing data.
        1. Estimated lines of code: Approximately 1,500 to 5,000 lines.

4. **Infrastructure and Configuration:**
    - Configuration files, deployment scripts, etc.: JavaScript code will be used for configuring the server environment, deploying the application, and managing infrastructure.
        1. Estimated lines of code: Approximately 1,000 to 5,000 lines.

Here's the revised breakdown of estimated lines of code for VacaNest with the backend entirely in JavaScript:

- Backend: Approximately 75,500 to 290,000 lines.
- Frontend: Approximately 33,000 to 150,000 lines.
- Database: Approximately 1,500 to 5,000 lines.
- Infrastructure and Configuration: Approximately 1000 to 5,000 lines.

Total lines of code: Roughly 110,000 to 450,000 lines.

LOC Calculation using Function Point Analysis

1. **External Inputs (EI):**
    - User registration form submissions: Average (4)
    - Property listing creation: Average (4)
    - Updating property availability calendar: Simple (3)
    - User feedback submission: Average (4)
    - Contact form submissions: Simple (3)
    - User login/authentication: Average (4)
    - Property review submissions: Average (4)
    - Property inquiry submissions: Average (4)
    - Property booking requests: Average (4)
    - Property rating submissions: Simple (3)
    - User profile creation/update: Average (4)
    - User password reset requests: Simple (3)

- Property photo uploads: Average (4)
- User account settings updates: Simple (3)
- Property pricing updates: Average (4)
- User review moderation requests: Average (4)
- Property amenity additions/updates: Average (4)
- User message sending/receiving: Average (4)
- User booking cancellation requests: Simple (3)
- Property description updates: Simple (3)
- Property listing update requests: Simple (3)
- User account deletion requests: Simple (3)
- User account suspension requests: Simple (3)
- Property listing deletion requests: Simple (3)
- Property booking modification requests: Simple (3)

2. **External Outputs (EO):**
    - Property search results display: Average (5)
    - Booking confirmation emails: Simple (4)
    - Property details pages: Average (5)
    - Review reminders for past guests: Simple (4)
    - Weekly/monthly booking summary reports: Average (5)
    - Property booking confirmation pages: Simple (4)
    - Property review display pages: Average (5)
    - User account summary pages: Average (5)
    - User notification emails: Simple (4)
    - Property booking cancellation confirmations: Simple (4)
    - Property availability calendars: Average (5)
    - User booking history pages: Average (5)
    - Property review summary reports: Average (5)
    - Admin activity summary reports: Average (5)
    - Property inquiry response emails: Simple (4)
    - User message notification emails: Simple (4)
    - Property booking modification confirmations: Simple (4)
    - User review moderation notifications: Simple (4)
    - Property pricing change notifications: Simple (4)
    - User account deletion confirmations: Simple (4)
    - Property listing update confirmations: Simple (4)
    - User account update confirmations: Simple (4)
    - Property listing deletion confirmations: Simple (4)

3. **External Inquiries (EQ):**
    - Property availability check based on dates: Average (4)
    - User search for properties within a specific location: Average (4)
    - Retrieving property amenities list: Simple (3)
    - Finding user contact information: Average (4)
    - Fetching booking status for a specific property: Simple (3)
    - User profile view by other users: Average (4)
    - Property availability calendar view by users: Average (4)
    - User message retrieval: Average (4)
    - Property review retrieval: Average (4)
    - User booking history retrieval: Average (4)
    - Property pricing inquiry: Average (4)
    - User review search by property: Average (4)
    - Admin log retrieval: Average (4)
    - Property inquiry response status check: Simple (3)
    - User notification status check: Simple (3)
    - Property booking modification status check: Simple (3)
    - User review moderation status check: Simple (3)
    - Property pricing change history retrieval: Simple (3)
    - User account deletion status check: Simple (3)
    - Property description retrieval by users: Simple (3)
    - Property listing update status check: Simple (3)
    - User account update status check: Simple (3)
    - Property listing deletion status check: Simple (3)

4. **Internal Files (ILF):**
    - User profiles and preferences: Average (10)
    - Property descriptions and images: Average (10)
    - Booking history and payment records: Average (10)
    - User reviews and ratings: Average (10)
    - Admin logs and activity history: Average (10)
    - Property availability calendars: Average (10)
    - User message history: Average (10)
    - Property booking records: Average (10)
    - User notification records: Simple (7)
    - Property review records: Average (10)
    - User login activity records: Average (10)
    - Property pricing history records: Average (10)

- User account activity logs: Average (10)
- Property inquiry response records: Average (10)
- User message delivery logs: Average (10)
- Property booking modification logs: Average (10)
- User review moderation logs: Simple (7)
- Property amenity records: Average (10)
- User account deletion logs: Simple (7)
- Property description history records: Simple (7)
- Property listing update logs: Simple (7)
- User account update logs: Simple (7)
- Property listing deletion logs: Simple (7)

5. **External Interfaces (EIF):**
    - Integration with social media platforms for login/authentication: Average (7)
    - API integration for weather forecasts for property locations: Average (7)
    - Integration with local event calendars for area attractions: Average (7)
    - Connection to third-party review platforms for guest feedback: Average (7)
    - Integration with property management systems for syncing availability and rates: Average (7)
    - Payment gateway integration for booking transactions: Average (7)
    - Integration with messaging services for user notifications: Average (7)
    - Integration with mapping services for property locations: Average (7)
    - Integration with customer support platforms for user inquiries: Average (7)
    - Integration with marketing platforms for promotional campaigns: Average (7)
    - Integration with analytics platforms for user behavior tracking: Average (7)
    - Integration with identity verification services for user authentication: Average (7)
    - Connection to regulatory compliance services for data protection: Simple (5)
    - Integration with payment processing services for refund transactions: Average (7)
    - Integration with translation services for multilingual support: Average (7)
    - Connection to advertising platforms for property promotion: Average (7)
    - Integration with CRM systems for customer relationship management: Average (7)

- Connection to inventory management systems for property listings: Average (7)
- Integration with review aggregation platforms for reputation management: Average (7)
- Connection to legal services for terms and conditions enforcement: Simple (5)
- Integration with accessibility tools for inclusive user experience: Average (7)
- Integration with email marketing services for user engagement: Average (7)
- Connection to reporting tools for data analysis and visualization: Average (7)

Classification of Functional Point:

Component	Simple	Average	Complex
EI (External Inputs)	3	4	6
EO (External Outputs)	4	5	7
EQ (External Inquiries)	3	4	6
EIF (External Interface Files)	5	7	10
ILF (Internal Logical Files)	7	10	15

Overall projects Functional Points:

Component	Simple	Average	Total Points
EI	12	13	25
EO	14	9	23
EQ	12	11	23
ILF	7	16	23
EIF	2	21	23

<u>Let's answer each question for the VacaNest system:</u>

1. Does the system require reliable backup and recovery?
   Score: 4 (Important)
2. Are data communications required?
   Score: 5 (Absolutely Essential)
3. Are there distributed processing functions?
   Score: 3 (Moderate)
4. Is performance critical?
   Score: 4 (Important)
5. Will the system run in an existing, heavily utilized operating environment?
   Score: 3 (Moderate)
6. Does the system require online data entry?
   Score: 5 (Absolutely Essential)
7. Does the online data entry require the input transaction to be built over multiple screens or operations?
   Score: 3 (Moderate)
8. Are the master files updated online?
   Score: 4 (Important)
9. Are the inputs, outputs, files, or inquiries complex?
   Score: 4 (Important)
10. Is the internal processing complex?
    Score: 3 (Moderate)
11. Is the code designed to be reusable?
    Score: 4 (Important)

12. Are conversion and installation included in the design?
    Score: 3 (Moderate)
13. Is the system designed for multiple installations in different organizations?
    Score: 3 (Moderate)
14. Is the application designed to facilitate change and ease of use by the user?
    Score: 5 (Absolutely Essential)

Now, let's sum the 14 complexity adjustment values:

ΣFi = 4 + 5 + 3 + 4 + 3 + 5 + 3 + 4 + 4 + 3 + 4 + 3 + 3 + 5
ΣFi = 55

Given:

- Total Count (Count-Total) = 647
- Sum of complexity adjustment values (ΣFi) = 55

Using the formula:

$FP=(Count-Total)\times[0.65+0.01\times\Sigma Fi]$

Substituting the values:

$FP=647\times[0.65+0.01\times55]$

FP=647×[0.65+0.01×55]

$FP=647\times[0.65+0.55]$

FP=647×[0.65+0.55]

$FP=647\times1.20$

FP=647×1.20

$FP=776.4$

FP=776.4

Rounded to the nearest whole number, the Functional Points (FP) for the VacaNest system is approximately 776.

Allocating Functional point to backend, frontend and database:

To appropriately distribute Functional Points (FP) between the frontend and backend components of the system, we consider the respective responsibilities and significance of each part. The frontend, tasked with rendering user interfaces, managing interactions, and facilitating communication with the backend, warrants a portion of the FP allocation.

A reasonable distribution strategy involves assigning approximately 30% of the total FP to the frontend, reflecting its critical role in ensuring user experience and interaction quality. Consequently, the remaining 70% is allocated to the backend to account for its extensive responsibilities in handling business logic, data processing, and server-side operations.

<u>For the frontend, the calculated FP allocation is as follows:</u>

FP for Frontend=0.3×Total FP=0.3×776≈233.8

<u>Similarly, for the backend, the calculated FP allocation is:</u>

*FP for Backend = 0.6 * Total FP = 0.6 * 776 ≈ 465.6*

<u>For the Database, the calculated FP allocation is as follows:</u>

*FP for Database = 0.1 * Total FP = 0.1 * 776 ≈ 77.6*

Now, to estimate the Lines of Code (LOC) for each component, we utilize the respective LOC per Function Point values:

- **For JavaScript (backend):**
- LOC per Function Point=58
- LOC for Backend = 465.6 * 58
- LOC for Backend ≈ 26988.8

- **For HTML (frontend):**
- LOC per Function Point=40
- LOC for Frontend = 232.8 * 40

- LOC for Frontend ≈ 9312

- **For Database:**
- LOC per Function Point=35
- LOC for Database = 77.6 * 35
- LOC for Database ≈ 2716

  Note:FP for Database * LOC per Function Point for JavaScript (assuming database code is JavaScript-based+SQL+PLSQL)

Total Lines of code:
Total LOC = LOC for Frontend + LOC for Backend + LOC for Database
Total LOC = 9312 + 26988.8 + 2716
Total LOC ≈ *39016.8 → 39K*

Understanding the Discrepancy in Estimated Lines of Code for VacaNest Platform

When estimating the lines of code (LOC) required for developing software systems like VacaNest, it's crucial to employ accurate and reliable methods to gauge the scale and complexity of the project. One common approach is the Functional Point (FP) analysis, which assesses the functionalities, inputs, outputs, inquiries, and internal files of the system to derive an estimation of LOC. However, the estimated LOC obtained from the FP analysis may sometimes differ significantly from rough estimations based on prior experience or other factors. This explanation aims to delve into the reasons behind the notable difference in estimated LOC and elucidate why platforms like VacaNest necessitate a substantial amount of code for development.

Factors Contributing to the Difference in Estimated LOC:

1. **Granularity of Analysis:**
   - The FP analysis entails a detailed examination of system functionalities, considering various aspects such as user inputs, outputs, inquiries, and internal files. This granular approach may lead to a more accurate estimation of LOC compared to rough estimations, which might overlook certain complexities.

2. **Complexity Adjustment:**
    - The FP analysis incorporates complexity adjustment factors based on General System Characteristics (GSC) such as data communications, response times, and end-user efficiency. These adjustments can significantly impact the final FP count and subsequently the estimated LOC.
3. **Scope Definition:**
    - The initial rough estimation may provide a broad overview of the project scope, whereas the FP analysis involves a more detailed definition of functionalities and requirements. As the analysis progresses, the scope may be refined, leading to a more precise estimation.

Why Platforms Like VacaNest Require Extensive Lines of Code:

1. **Feature-rich Functionality:**
    - Platforms like VacaNest encompass a wide array of features and functionalities, including user registration, property listing management, search and booking capabilities, user interactions, payments, notifications, and administrative tools. Each of these functionalities requires substantial code to implement and integrate seamlessly.
2. **Scalability and Performance:**
    - To accommodate the high volume of users, transactions, and data associated with platforms like VacaNest/Airbnb, the software architecture must be scalable and optimized for performance. This necessitates additional code for implementing efficient data structures, caching mechanisms, load balancing, and optimization techniques.
3. **Security Considerations:**
    - Security is paramount for platforms handling sensitive user data and financial transactions. Implementing robust security measures, such as encryption, authentication, authorization, and secure communication protocols, requires extensive code to fortify the system against vulnerabilities and threats.
4. **Complex Business Logic:**
    - The intricate business logic underlying platforms like VacaNest, including pricing algorithms, recommendation engines, booking workflows, payment processing, and dispute resolution mechanisms, entails a considerable

amount of code to ensure accuracy, reliability, and compliance with business requirements.

Conclusion: In conclusion, the discrepancy in estimated Lines of Code between rough estimations and Functional Point analysis results from differences in granularity, complexity adjustment, and scope definition. Platforms like VacaNest demand extensive lines of code due to their feature-rich functionality, scalability requirements, security considerations, and complex business logic. While the estimated LOC may vary depending on the estimation method used, it's essential to recognize the inherent complexities of developing large-scale software systems and employ rigorous analysis techniques to derive accurate estimations.

1. Functional Point Analysis (FPA) LOC:

- This estimate is derived from a detailed analysis of system functionalities, inputs, outputs, inquiries, and internal files. It takes into account the specific requirements and complexities of the project. Using this estimate provides a more accurate reflection of the actual code needed based on the functional scope of the project.

2. Rough Estimation LOC:

- This estimate is based on prior experience, industry benchmarks, and general assumptions about the project's size and complexity. It provides a high-level overview and may not capture all the nuances and intricacies of the project requirements. However, it can serve as a quick approximation in the early stages of planning.

3. Consider a Middle Ground:

"By finding a middle ground between these two approaches, we aim to strike a balance between precision and efficiency. This middle ground approach allows us to leverage the insights from both methods while mitigating the limitations of each individual estimation approach."

- First, let's summarize the estimates for LOC based on Functional Point Analysis (FPA) and rough estimation:

1. Functional Point Analysis (FPA) LOC:
    - Backend: Approximately 26,988 LOC
    - Frontend: Approximately 9,312 LOC
    - Database: Approximately 2,716 LOC
    - Total: Approximately 39,016 LOC

2. Rough Estimation LOC range:
    - Total: Roughly 110,000 to 450,000 LOC

Now, to find the middle ground estimate, we can take the midpoint of the rough estimation range and calculate the average between that and the FPA-based LOC:

Midpoint of rough estimation range:

- *110,000+450,000 / 2 = 280,000 LOC*

Average LOC:

- 39,000+280,000 = 319,000
- 319,000 / 2 = 159,500 LOC

Therefore, the middle ground estimate for the Lines of Code (LOC) required for the VacaNest system is approximately ≈ ***159,500 →159K***

152,758 LOC. This approach balances the detailed analysis provided by Functional Point Analysis with the high-level estimation provided by the rough estimation method.

Effort Estimations Through COCOMO I Basic Model

The VacaNest software project falls into the *semi-detached* category in COCOMO due to its moderate size of 159K LOC and complexity, mixed team composition, flexibility for changing requirements, and scale that is substantial but not as extensive as projects categorized as embedded.

Given:

- a (Scale Factor): 3.0
- b (Exponent Factor): 1.12
- c (Scale Factor for exponent): 2.5
- d (Product Rating): 0.35
- KLOC: 159

Effort:

$$Effort = a \times (KLOC)^{b}$$
$$Effort = 3.0 \times (159)^{1.12}$$
$$Effort \approx \mathbf{876.36\ Person\ Month}$$

Development Time:

$$Development\ Time = c \times (Effort)^{d}$$
$$Development\ Time = 2.5 \times (876.36)^{0.35}$$
$$Development\ Time \approx \mathbf{26.7\ months}$$

Staff Size:

$$Staff\ \text{Size} = Effort\ /\ Development\ Time$$
$$Staff\ \text{Size} = 876.36\ /\ 26.7$$
$$Staff\ \textbf{Size = 32.8 staff member on Average}$$

Productivity:

$$Productivity = LOC \,/\, Effort$$
$$Productivity = 159{,}000 \,/\, 876.36$$
$$Productivity \approx \mathbf{181.4\ loc/Person\text{-}month}$$

Effort Estimations Through COCOMO I Intermediate Model

The basic Cocomo model assumes that the effort is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system's effort and schedule can be solely calculated based on Lines of Code. For that,and various other factors such as reliability, experience, and Capability. These factors below are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation. Classification of Cost Drivers and their Attributes:

1. Required software reliability extent: High →1.15
2. Size of the application database: Very High →1.16
3. Complexity of the product: High →1.15
4. Run-time performance constraints: High →1.11
5. Memory constraints: High →1.06
6. Volatility of the virtual machine environment: Low →0.87
7. Required turnabout time: Nominal →1
8. Analyst capability: High →0.86
9. Software engineering capability: High →0.86
10. Application experience: Very High →0.82
11. Virtual machine experience: Low →1.10
12. Programming language experience: High →0.95
13. Use of software tools: High →0.91
14. Application of software engineering methods: High →0.91
15. Required development schedule: Nominal →1

Given:

- a (Scale Factor): 3.0
- b (Exponent Factor): 1.12
- c (Scale Factor for exponent): 2.5

- d (Product Rating): 0.35
- KLOC: 159
- EAF: 0.82416

Effort:

$$Effort = a \times (KLOC)^{b} \times \text{EAF}$$
$$Effort = 3.0 \times (159)^{1.12} \times 0.824$$
$$Effort \approx \mathbf{722.26\ Person\ Month}$$

Development Time:

$$Development\ Time = c \times (Effort)^{d}$$
$$Development\ Time = 2.5 \times (722.26)^{0.35}$$
$$Development\ Time \approx \mathbf{25.03\ months}$$

Staff Size:

$$Staff\ \text{Size} = Effort\ /\ Development\ Time$$
$$Staff\ \text{Size} = 722.26\ /\ 25.03$$
$$Staff\ \mathbf{Size = 28.8\ staff\ member\ on\ Average}$$

Productivity:

$$Productivity = LOC\ /\ \text{Effort}$$
$$\text{Productivity} = 159{,}000\ /722.26$$
$$Productivity \approx \mathbf{220.14\ loc/Person\text{-}month}$$

Effort Estimations Through COCOMO I Detailed Model

The Detailed COCOMO takes the effort estimations obtained from the intermediate model and refines them further by considering the specific phases of the project lifecycle. In our case, where we're developing a comprehensive platform like VacaNest, falling into the semi-detached large category with approximately 159K lines of code, the Detailed COCOMO model becomes essential for precise planning and resource allocation. By breaking down the effort and development time into distinct phases such as planning, requirements, system design, detailed design, module coding, integration, and testing, the Detailed COCOMO model provides a more granular understanding of resource requirements and project milestones. This approach enables project managers to allocate manpower efficiently and schedule activities effectively, ensuring smooth progress throughout the development lifecycle.

Effort:

$$Effort = a \times (KLOC)^{b} \times \text{EAF}$$
$$Effort = 3.0 \times (159)^{1.12} \times 0.824$$
$$Effort \approx \textbf{722.26 Person Month}$$

Development Time:

$$Development\ Time = c \times (Effort)^{d}$$
$$Development\ Time = 2.5 \times (722.26)^{0.35}$$
$$Development\ Time \approx \textbf{25.03 months}$$

Effort Required for different phases of the project:

Phases	**μp × Effort**	**Required effort**
Plan & Requirements	0.07 × 722.26	50.55 PM
System Design	0.17 × 722.26	122.78 PM
Detailed Design	0.24 × 722.26	173.34 PM
Module Code & Test	0.31 × 722.26	223.9 PM
Integration & Test	0.28 × 722.26	202.23 PM

Development Time Required for different phases of the project:

Phases	**Ƭp × Development-Time**	**Required Development-Time**
Plan & Requirements	0.22 × 25.03	5.5 Months
System Design	0.27 × 25.03	6.7 Months

Phases	**Ƭp × Development-Time**	**Required Development-Time**
Detailed Design	0.19 × 25.03	4.7 Months
Module Code & Test	0.25 × 25.03	6.2 Months
Integration & Test	0.29 × 25.03	7.2 Months

<u>Estimates Through COCOMO II Model</u>

<u>1.Application composition Model</u>

Screens = External Inputs, External Outputs and External Queries
Reports = External Outputs and External Queries
3GL components= Internal Logical Files and External Interface Files

1. Screens:
   Server-Side Screens: 25 EI + 23 EQ = 48
   Client-Side Screens: 23 EO = 23
2. Reports:
   Server-Side Reports: 23 EQ = 23
   Client-Side Reports: 23 EO = 23
3. 3GL Components:
   Server-Side 3GL Components: 23 ILF + 23 EIF = 46
   Client-Side 3GL Components: None

components	value	server	client
screens	71	48	23
reports	46	23	23

components	**no.of views**	**no.of sources**	**category**	**complexity**	**Object points**
Screens	>8	8+	difficult	2 x 71	142
Reports	4+	8+	difficult	8 x 46	368
3GL components	-	-	-	10 x 46	460
Total object points					**970**

New object Point:

Certain components can be readily reused from external sources:

External Inputs (EI):
Hypothetical Reuse Percentage: 0%
Calculation: Out of 25 EI components,(0% of 25)
External Outputs (EO):
Hypothetical Reuse Percentage: 5%
Calculation: Out of 23 EO components(5% of 23) reused.
External Inquiries (EQ):
Hypothetical Reuse Percentage: 5%
Calculation: Out of 23 EQ components, (5% of 23) reused.
Internal Logical Files (ILF):
Hypothetical Reuse Percentage: 10%
Calculation: Out of 23 ILF components, (10% of 23)reused.
External Interface Files (EIF):
Hypothetical Reuse Percentage: 5%
Calculation: Out of 23 EIF components(5% of 23) reused.

Sum up the individual reuse percentages:
- EI: 0%
- EO: 5%

- EQ: 5%
- ILF: 0%
- EIF: 5%

Total reuse percentage =0% + 5% + 5% + 0% + 5% = 10%

NOP = (object points) ×(100 - %reuse)/ 100
NOP= (970) x(100- 10)/100
NOP= 873

Effort:

Productivity Rate: (Taking a nominal value= 13)
NOP :873
Effort= NOP/Productivity-rate
Effort= 873/4
Effort= 218.5 PM

2.Early design:

A = constant representing the nominal productivity where A = 2.5
B = Scaling Factors
Size = 159k

Calculating B:
Precedence:(experience of similar project) = low(4.96)
Development flexibility: (degree of flexibility)= low(4.05)
Architecture Risk and Resolution:(risk analysis)= high(2.85)
Team cohesion:(Team management skills)= average(3.29)
Process Maturity: (maturity of organization) = high(3.12)

B = 0.91 + 0.01 * (Sum of rating scaling factors for project)
B= 0.91+0.01*(4.96+4.05+2.85+3.29+3.12)
B=1.09
B > 1.0, rate of increase of effort increase as the size of product is increased

Effort For the Project:

PM Nominal:

$$PM=A\times(size)^{B}$$
$$PM= 2.5\times(159)^{1.09}$$
$$PM \approx 627.28$$

PM Adjusted:

seven effort multipliers in the Early Design model:

1. **Precedentedness (PREC): Nominal (1.00)**
   The project includes a mix of average and simple external inputs, outputs, inquiries, and internal files, indicating a typical level of familiarity with previous projects.
2. **Development Flexibility (FLEX): High (1.07)**
   The project involves various types of inputs, outputs, inquiries, and internal files, suggesting a need for some degree of flexibility in development.
3. **Risk Resolution (RESL): Nominal (1.00)**
   The project has a mix of average and simple external interfaces, indicating a typical level of risk resolution capability.
4. **Team Cohesion (TEAM): Nominal (1.00)**
   The project involves a mix of average and simple external interfaces and internal files, suggesting a typical level of team cohesion.
5. **Process Maturity (PMAT): Nominal (1.00)**
   The project includes a mix of average and simple external interfaces and internal files, indicating a typical level of process maturity.
6. **Required Reusability (RELY): Nominal (1.00)**
   The project does not have specific requirements for high or low re-usability, suggesting a typical level of required re usability.
7. **Architecture/Risk Resolution (ARCH): High (1.07)**
   The project involves various types of inputs, outputs, inquiries, and internal files, suggesting a need for some degree of architecture and risk resolution capability.

$$PM=A\times(size)^{B} \text{ x } \sum \text{EMi(7 total)}$$
$$PM= 2.5\times(159)^{1.09}\text{x } 7.14$$
$$PM= 4478.79$$

3- Post architecture:

Post Architecture model effort multipliers:

1- Precedentedness (PREC): Nominal (1.00)
The project includes a mix of average and simple external inputs, outputs, inquiries, and internal files, indicating a typical level of familiarity with previous projects.

2- Development Flexibility (FLEX): High (1.07)
The project involves various types of inputs, outputs, inquiries, and internal files, suggesting a need for some degree of flexibility in development.

3- Risk Resolution (RESL): Nominal (1.00)
The project has a mix of average and simple external interfaces, indicating a typical level of risk resolution capability.

4- Team Cohesion (TEAM): Nominal (1.00)
The project involves a mix of average and simple external interfaces and internal files, suggesting a typical level of team cohesion.

5- Process Maturity (PMAT): Nominal (1.00)
The project includes a mix of average and simple external interfaces and internal files, indicating a typical level of process maturity.

6- Required Reusability (RELY): Nominal (1.00)
The project does not have specific requirements for high or low reusability, suggesting a typical level of required reusability.

7- Architecture/Risk Resolution (ARCH): High (1.07)
The project involves various types of inputs, outputs, inquiries, and internal files, suggesting a need for some degree of architecture and risk resolution capability.

8- Analyst Capability (ACAP): Nominal (1.00)

The project involves a mix of average and simple external interfaces and internal files, suggesting a typical level of analyst capability.

9- Programmer Capability (PCAP): Nominal (1.00)
The project involves a mix of average and simple external interfaces and internal files, suggesting a typical level of programmer capability.

10- Personnel Continuity (PCON): Nominal (1.00)
The project does not have specific requirements for high or low personnel continuity, suggesting a typical level of continuity.

11- Application Experience (APEX): Nominal (1.00)
The project involves a mix of average and simple external interfaces and internal files, suggesting a typical level of application experience.

12- Platform Experience (PLEX): Nominal (1.00)
The project involves a mix of average and simple external interfaces and internal files, suggesting a typical level of platform experience.

13- Language and Tool Experience (LTEX): Nominal (1.00)
The project involves a mix of average and simple external interfaces and internal files, suggesting a typical level of language and tool experience.

14- Use of Software Tools (TOOL): Nominal (1.00)
The project involves a mix of average and simple external interfaces and internal files, suggesting a typical level of use of software tools.

15- Multi site Development (SITE): Nominal (1.00)
The project does not involve specific requirements for multi site development, suggesting a typical development environment.

16- Required Development Schedule (SCED): Nominal (1.00)
The project does not have specific requirements for high or low schedule constraints, suggesting a typical development schedule.

17- Extent of Documentation (DOCU): Nominal (1.00)

The project involves a mix of average and simple external interfaces and internal files, suggesting a typical level of documentation extent.

Total of effort multipliers:

sum= 1+1+1.07+1+1+1+1.07+1+1+1+1+1+1+1+1+1+1

sum= 17.14

PM=A×(size)^{B} x ∑ EM$_{I}$(17 total)

PM= 2.5×(159)$^{1.09}$x 17.14

PM= 10751.61

Time of development:

TDEV=(3.67 x (PM)$^{0.28 + 0.2 \times (B-1.01))}$)x(SCED%/100)

TDEV=(3.67 x (672.28)$^{0.28 + 0.2 \times (1.09-1.01))}$)x(100/100)

TDEV= 25.21months

REUSE Model:

1- Design Modified (DM):

- Extensive feature-rich functionality necessitates substantial architectural adjustments.
- Robust security measures and complex business logic demand significant design modifications.

2- Code Modified (CM):

- Implementation of scalable, performant code and robust security measures requires extensive code modifications.
- Handling complex business logic entails a considerable amount of additional code.

3-Integrated Modified (IM):

- Integration efforts, while crucial, may be relatively straightforward compared to design and code modifications.
- While critical, integration does not demand as much effort as other phases.

4- Software Understanding:

- Reasonably well structured; some weak areas
- Moderate correlation between program and application.
- Moderate level of code commentary, headers, documentations**.**
  So we are taking nom 30.

5-Assessment and Assimilation:

- Considerable Module testing and evaluation and documentation so We have 6 values for this.

6- UNFM:

- Somewhat familiar so we taking it 0.4

Given:

- CM: 40%
- IM: 20%
- DM: 40%
- SU : Nom 30
- AA: 6
- UNFM: 0.4

  Solution:
  AAF= 0.4 x DM% +0.3 x CM% + 0.3 x IM%
  AAF= 0.4 x 40% + 0.3 x 40% + 0.3 x 20%
  AAF= 0.34

  ESLOC= (ASLOC(AA+AAF(1+0.02(SU)(UNFM))) /100
  ESLOC= (159000 (6+0.34(1+0.02(30)(0.4))) /100
  ESLOC= 10210.34

  Effort = A x $(ESLOC)^B$
  Effort = 2.5 x $(10210.34)^{0.19}$
  Effort = 14.44 PM

COSYSMO :

There are 14 cost drivers:

Sure, let's reassess the scale values for each cost driver based on the project's characteristics:

1. Requirement Understanding: High (H)

- A diverse range of inputs, outputs, inquiries, files, and interfaces, indicating a need for a high level of understanding to accurately capture and address requirements and minimize potential misunderstandings or rework.

2. Architecture Understanding: High (H)

- The complexity and interdependence of external inputs, outputs, inquiries, files, and interfaces necessitate a deep understanding of the project's architecture to design robust systems.

3. Level of Service Requirement: Very low (VL)

- The descriptions of external inputs, outputs, inquiries, files, and interfaces suggest standard service requirements without extraordinary demands or specialized features.

4. Migration Complexity: Nominal (N)

-Integration with multiple external interfaces indicates potential complexities in data migration, system integration, and compatibility, necessitating a higher effort for migration tasks to ensure smooth transition and system operability.

5.Technology Risk: Nominal (N)

- Integration with diverse external systems poses technology risks such as compatibility issues and data security concerns, requiring proactive risk management efforts and potentially increasing project effort to mitigate these risks.

6. Documentation: Low (L)

- The standard descriptions of inputs, outputs, inquiries, files, and interfaces suggest a typical level of documentation effort for clarity and maintainability, with no indication of particularly extensive or minimal documentation needs.

7.Delivery of Installation and Platforms: Nominal (N)

Integration with various external interfaces and the need for seamless deployment suggest additional effort may be required for delivery, installation, configuration, and platform support tasks to ensure successful implementation and system functionality.

8.Recursive Level in the Design: Low (L)
- The complexity of external inputs, outputs, inquiries, files, and interfaces implies a need for a higher level of understanding and effort in design tasks to address inter dependencies and ensure system scalability and flexibility.

9. Stakeholder Team Cohesion: Very High (VH)
- The complexity and diversity of project elements necessitate strong cohesion among stakeholders for effective communication, collaboration, and alignment of goals to minimize misunderstandings and ensure project success.

10. Personnel Team Capability: High (H)
- The varied requirements and potential complexities in external inputs, outputs, inquiries, files, and interfaces suggest a need for capable team members with diverse skills and expertise to handle tasks effectively and deliver quality outcomes.

11. Personnel Experience Continuity: High (H)
- Essential to leverage existing knowledge and expertise, minimize ramp-up time, and ensure consistent project execution, especially considering the project's complexity and potential challenges.

12. Process Capability: Extra High (EH)
- Ensure efficient project execution, maintain quality standards, and manage project risks effectively throughout the project life cycle.

13.Multi-site Coordination: High (H)
- While the project involves various inputs, outputs, inquiries, files, and interfaces, there is no explicit indication of coordination challenges across multiple sites. Therefore, a nominal scale value is more appropriate as there may be some coordination efforts, but they are not expected to be exceptionally complex or demanding.

14.Tool Support: Very High (VH)
- Adequate tool support is essential to streamline development processes, automate repetitive tasks, and facilitate collaboration among team members working on various external inputs, outputs, inquiries, files, and interfaces, enhancing project efficiency and

productivity.

Solution:

Cost drivers:
0.77+0.81+0.62+1+1+0.88+0.88+0.87+0.65+0.81+0.82+0.68+0.72+0.72
Total = 11.23
A= calibration constant derived from historical project data= 1 there is no existing project
E= economy or diseconomy of scale taking default value =1.0

PMns= A x(Size)^{E} xi=0$^{\Pi n}$ EMi
PMns= 1 x (159)$^{1.0}$ x 11.23
PMns= 1785.57

Expert Judgment with Delphi Method:

The application of the Delphi Method in the estimation process for the VacaNest project yielded valuable insights and consensus among the expert panel. Here are the results of the Delphi Method:

1. **Project Size:**
    - The consensus among experts indicates that VacaNest is a large -scale project, with estimates ranging from medium to large-sized.
    - Expert 1 estimated the project size at around 150,000 lines of code (LOC), emphasizing extensive features.
    - Expert 2 categorized the project as medium to large-sized, prioritizing features based on business value and complexity.
    - Expert 3 assessed the project size based on anticipated functionalities and complexity factors.

2. **Complexity:**
    - All experts agreed on the high complexity of the VacaNest project due to its multi-layered technology stack, intricate business logic, and seamless user interactions.
    - Challenges identified include technology integration, security concerns, and optimization of user experience.

3. **Resource Requirements**:
    - Essential resources identified by experts include frontend and backend developers, UI/UX designers, testers, and possibly DevOps engineers.
    - Allocation of resources is suggested based on project phases and critical path analysis to ensure efficient utilization.

4. **Risks:**
    - Identified risks encompass technical challenges, regulatory changes, scalability issues, and resource constraints.
    - Mitigation strategies proposed include thorough testing, contingency planning, and stakeholder engagement to minimize disruptions.

5. **Constraints:**
    - Budget limitations, tight timelines, and technology compatibility issues were recognized as constraints that could impact project estimation.
    - Proposed approaches for addressing constraints include prioritization of features, negotiation with stakeholders, and leveraging existing frameworks.

Process of the Delphi Method:

The Delphi Method involves the following steps in the estimation process:

1. **Initial Questionnaire:** Experts are invited to participate in the estimation process by providing their opinions and insights through an initial questionnaire. This questionnaire is designed to gather expert opinions on various aspects of the project, including size, complexity, resource requirements, risks, and constraints.
2. **Anonymized Feedback:** Expert responses are anonymized to ensure impartiality and privacy. This encourages experts to express their opinions freely without fear of bias.
3. **Iterative Feedback Loops:** Subsequent rounds of feedback and consensus-building are conducted to refine the estimation process. Experts review and provide feedback on the responses of their peers, leading to convergence towards a consensus.
4. **Confidentiality:** Confidentiality is maintained throughout the process to foster open communication and prevent bias. All data collected are anonymized to ensure impartiality and privacy.

5. **Informed Decision-making:** By leveraging the collective expertise of the panel, the Delphi Method facilitates informed decision-making in various aspects of the project. Consensus among experts informs decisions on size estimation, complexity assessment, resource requirements, risk identification, and constraint mitigation.
6. **Adaptability:** The Delphi Method allows for the adaptation of estimation techniques and models to suit the unique characteristics of the project. Experts draw from their experience and expertise to tailor their responses accordingly.
7. **Transparent Communication:** A robust feedback mechanism ensures transparent communication and timely decision-making. Stakeholder engagement is prioritized throughout the estimation process to ensure alignment with project goals and objectives.

Overall, the Delphi Method provides a structured framework for soliciting and synthesizing expert judgment to inform the estimation process. By harnessing the collective wisdom of experts, it facilitates consensus-building, mitigates bias, and enables informed decision-making, ultimately contributing to the success of the project.

<u>For Further Reference:</u>

For a comprehensive understanding of the Expert Judgment by Delphi Method employed in the estimation process, please refer to the accompanying document titled "Expert Judgment by Delphi Method." This document provides a detailed study on the methodology, process, and outcomes of utilizing the Delphi Method in the estimation of the VacaNest project. It offers insights into the structure of the Delphi Method, steps involved in the estimation process, anonymized feedback from participating experts, and a thorough analysis of the results obtained.

The document serves as a valuable resource for stakeholders, project managers, and researchers interested in exploring the efficacy of expert judgment techniques in project estimation and decision-making processes. It provides in-depth insights into the application of the Delphi Method in the context of software development projects, highlighting its strengths, limitations, and practical implications.

Agile Estimation Techniques:

We are applying Poker planning estimation technique in our project that encourages team collaboration and helps in achieving consensus on the effort required for tasks.

Poker Planning:

1.Preparation:

- Ensure that the team understands the user stories or backlog items to be estimated.
- Each item should be broken down into manageable pieces, typically in the form of user stories.

2. Estimation Session:

- Gather the team in a meeting or a virtual session.
- Explain the rules of Planning Poker: each team member will independently estimate the effort required for each item using a set of predefined values (usually Fibonacci sequence or t-shirt sizes).
- Each team member selects a card representing their estimate without revealing it to others.

3. Reveal and Discussion:

- Once everyone has made their estimate, all team members reveal their cards simultaneously.
- If there's a wide range of estimates, team members discuss their reasoning behind their estimates. This discussion helps in understanding different perspectives and aligning everyone's understanding of the task.

4. Consensus:

- After discussion, team members repeat the estimation process, usually for a few rounds, until a consensus is reached.

- The goal is not necessarily to achieve complete agreement on the estimate but rather to come to a shared understanding of the effort involved.

5. Recording Estimates:

- Record the final estimates for each item, usually in a shared document or agile tool.
- These estimates will help in prioritizing and planning the work in upcoming iterations.

6. Review and Retrospective:

- After completing the estimation session, it's beneficial to reflect on the process during the team's retrospective meeting.
- Discuss what went well and what could be improved for future estimation sessions.

Planning Poker fosters team collaboration, reduces bias, and encourages active participation from all team members. It's a versatile technique that can be adapted to various team sizes and project contexts.

Our project Estimates:

Of course, let's go through each category and estimate the effort for each item using Planning Poker.

- User registration form submissions: 5]
- Property listing creation: 5
- Updating property availability calendar: 3
- User feedback submission: 5
- Contact form submissions: 3
- User login/authentication:5

- Property review submissions: 5

- Property inquiry submissions: 5

- Property booking requests: 5

- Property rating submissions: 3

- User profile creation/update: 5

- User password reset requests: 3

- Property photo uploads:5

- User account settings updates: 3

- Property pricing updates: 5

- User review moderation requests: 5

- Property amenity additions/updates: 5

- User message sending/receiving: 5

- User booking cancellation requests: 3

- Property description updates: 3

- Property listing update requests: 3

- User account deletion requests: 3

- User account suspension requests: 3

- Property listing deletion requests: 3

- Property booking modification requests: 3

- Property search results display: 5

- Booking confirmation emails: 3

- Property details pages:5

- Review reminders for past guests: 3

- Weekly/monthly booking summary reports: 5

- Property booking confirmation pages: 3

- Property review display pages: 5

- User account summary pages: 5

- User notification emails: 3

- Property booking cancellation confirmations: 3

- Property availability calendars: 5

- User booking history pages: 5

- Property review summary reports: 5

- Admin activity summary reports: 5

- Property inquiry response emails: 3

- User message notification emails: 3

- Property booking modification confirmations: 3

- User review moderation notifications: 3

- Property pricing change notifications: 3

- User account deletion confirmations: 3

- Property listing update confirmations: 3

- User account update confirmations: 3

- Property listing deletion confirmations: 3

- Property availability check based on dates: 5

- User search for properties within a specific location: 5

- Retrieving property amenities list: 3

- Finding user contact information: 5

- Fetching booking status for a specific property: 3

- User profile view by other users: 5

- Property availability calendar view by users: 5

- User message retrieval: 5

- Property review retrieval: 5

- User booking history retrieval: 5

- Property pricing inquiry: 5

- User review search by property: 5

- Admin log retrieval: 5

- Property inquiry response status check: 3

- User notification status check: 3

- Property booking modification status check: 3

- User review moderation status check: 3

- Property pricing change history retrieval:3

- User account deletion status check: 3

- Property description retrieval by users: 3

- Property listing update status check: 3

- User account update status check: 3

- Property listing deletion status check: 3

- User profiles and preferences: 13

- Property descriptions and images: 13

- Booking history and payment records: 13

- User reviews and ratings:13

- Admin logs and activity history: 13

- Property availability calendars: 13

- User message history: 13

- Property booking records: 13

- User notification records: 8-
  Property review records: 13

- User login activity records: 13

- Property pricing history records: 13

- User account activity logs: 13

- Property inquiry response records: 13

- User message delivery logs: 13

- Property booking modification logs: 13

- User review moderation logs: 8

- Property amenity records: 13

- User account deletion logs: 8

- Property description history records: 8

- Property listing update logs: 8

- User account update logs: 8

- Property listing deletion logs: 8

External Interfaces (EIF):

- Integration with social media platforms for login/authentication: 8

- API integration for weather forecasts for property locations: 8

- Integration with local event calendars for area attractions:8

- Connection to third-party review platforms for guest feedback: 8

- Integration with property management systems for syncing availability and rates: 8

- Payment gateway integration for booking transactions: 8- Integration with messaging services for user notifications: 8

- Integration with mapping services for property locations: 8- Integration with customer support platforms for user inquiries: 8

- Integration with marketing platforms for promotional campaigns: 8

- Integration with analytics platforms for user behavior tracking: 8

- Integration with identity verification services for user authentication: 8

- Connection to regulatory compliance services for data protection: 5

- Integration with payment processing services for refund transactions: 8

- Integration with translation services for multilingual support: 8

- Connection to advertising platforms for property promotion: 8

- Integration with CRM systems for customer relationship management: 8

- Connection to inventory management systems for property listings: 8

- Integration with review aggregation platforms for reputation management: 8

- Connection to legal services for terms and conditions enforcement: 5

- Integration with accessibility tools for inclusive user experience:8

- Integration with email marketing services for user engagement: 8

- Connection to reporting tools for data analysis and visualization: 8

These estimates give a sense of the relative complexity and effort required for each task within each category.

Summary

The report delves into an in-depth analysis of various software cost estimation techniques as applied to the development of VacaNest, an envisioned vacation rental and property buying platform. By employing established methodologies including Lines of Code (LOC) calculation, Function Point Analysis (FPA), COCOMO, and COSYSMO, alongside innovative approaches such as Expert Judgment with Delphi Method and Agile Estimation Techniques, the aim is to offer a comprehensive understanding of project estimation.

Throughout the exploration, the report emphasizes the importance of accurate resource allocation, informed decision-making, and effective project management within the dynamic landscape of software development. By leveraging these methodologies, stakeholders can better anticipate project timelines, budgetary requirements, and resource needs, thereby enhancing the overall success potential of the VacaNest project.

Furthermore, the report underscores the significance of adapting estimation techniques to suit the unique requirements and challenges of the project. It emphasizes the iterative nature of estimation, acknowledging that adjustments and refinements may be necessary as the project progresses and new information becomes available.

In conclusion, the report serves as a valuable resource for project managers, developers, and stakeholders involved in the development of VacaNest, offering a roadmap for achieving accurate and reliable software cost estimation in order to facilitate successful project outcomes.