UNITED INTERNATIONAL UNIVERSITY

Project Report on Innovest

Course Title: Project Management
Course Code: PMG 4101
Section: A

Submitted to:

Kazi Abdun Noor
Lecturer
Department of CSE
United International University

Submitted by:

Meet our Team: (Group – 2)

Name:	**Student ID:**
Md. Shazzad Hossain Jiku	011221073
Md. Al- Emran	011221088
Md. Mohin Khan	011221091
Ananna Saha	011221117
Md. Ammar Hossain	011221601

Table of Contents

Introduction

1.1. Project Description

Innovest is an innovative web-based platform designed to bridge the gap between emerging startups and potential investors. The platform acts as a dynamic investment hub where aspiring entrepreneurs (fundraisers) can showcase their business ideas and seek funding, while investors can explore promising opportunities and invest in projects that align with their interests and risk appetite.

The platform is developed using modern web technologies, incorporating intuitive UI/UX, robust backend systems, and secure database integration. Innovest introduces a transparent and interactive environment where fundraisers can submit and manage ideas, communicate with interested investors, and track funding progress. Meanwhile, investors have access to a variety of vetted startup proposals, with features like investment tracking, and proposal management. Overall, Innovest is built with scalability, transparency, and ease of use in mind offering a unique solution to empower entrepreneurs and provide investors with diverse, secure, and profitable investment opportunities.

1.1. Motivation Behind the Project

The primary motivation behind Innovest stems from the growing need to support budding entrepreneurs who often struggle to secure funding for their innovative ideas. While many startups possess immense potential, the lack of financial backing and a proper platform to connect with investors often becomes a barrier to their growth. On the other hand, numerous individuals and organizations are willing to invest in promising ventures but face difficulties in finding credible, early-stage startups to support.

We observed a gap in the ecosystem where fundraisers and investors operate in silos with limited communication and collaboration opportunities. Innovest aims to fill that gap by creating a unified digital platform that empowers startups to present their ideas and seek funding while offering investors a secure, structured, and transparent environment to explore and invest in projects of their interest.

1.2. Vision

- Focus on supporting early-stage ideas and promoting economic innovation.

- Building a global network of changemakers, investors, and innovators.
- Empowering startups to turn ideas into impactful realities.
- Making investment decisions easier, smarter, and data-driven.
- Reducing barriers to entry in the funding world through accessible technology.
- Bridging the gap between visionary founders and modern investment tools.
- Driving a culture of innovation and entrepreneurship in emerging markets.

1.3. Scope and Features

- **Fundraiser Dashboard** – Idea submission, idea management, funding progress
- **Investor Panel** – Browse ideas, manage proposals, invest in project
- **Secure Payment Gateway** – bKash, Nagad & bank payment with installment options
- **Event Management** – Display of startup events, pitch sessions, and webinars
- **User Reviews & Ratings** – Feedback system for fundraisers and projects
- **Profile Management** – Role-based profile features for customization and tracking
- **Progress Tracking** – Visual tracking of investment status and growth
- **System Testing & Documentation** – Ensure platform quality, maintain future usability

Methodology

2.1. What is Methodology?

In the realm of software development, methodology refers to a well-defined framework or approach that governs the entire lifecycle of software creation. It outlines a set of practices, principles, and procedures used to plan, design, develop, test, and maintain software systems effectively. A methodology can consist of a single method or a combination of multiple methods, tailored to achieve specific project goals efficiently. Its primary objective is to enhance productivity, maintain quality standards, and ensure the final product meets client or stakeholder requirements.

Over the years, numerous software development methodologies have emerged, each offering unique benefits and challenges. Below are six widely used methodologies that are commonly adopted in software development projects:

- Waterfall Model

- Spiral Model
- Agile Model
- Scrum Model
- Extreme Programming XP Model
- Incremental Model

2.2. Waterfall Model

The Waterfall Model was the first Process Model to be introduced. It illustrates the development process in a linear sequential flow. That's why, It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

2.2.1. Waterfall Model Design

Here are the sequential phases of Waterfall model:

1. **Requirement Gathering:** All possible requirements of the project are gathered in this phase and documented in a requirement specification document.
2. **Design:** With the requirements established, the project team develops a detailed plan and prepares the design. This design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
3. **Implementation:** With inputs from the design, The development team starts creating the project according to the established plan, completing tasks step by step in sequence.
4. **Testing:** Once development is finished, the project undergoes thorough testing to ensure it meets the expected standards. If any issues or bugs found during testing are resolved in this phase.
5. **Deployment:** After successful testing and approval of a project is done, the product is deployed in the customer environment or released into the market.
6. **Maintenance:** After launch, the project enters a maintenance phase. There are some issues which come up in the client environment. To fix those issues,

patches are released. Also to enhance the product some better versions are released.

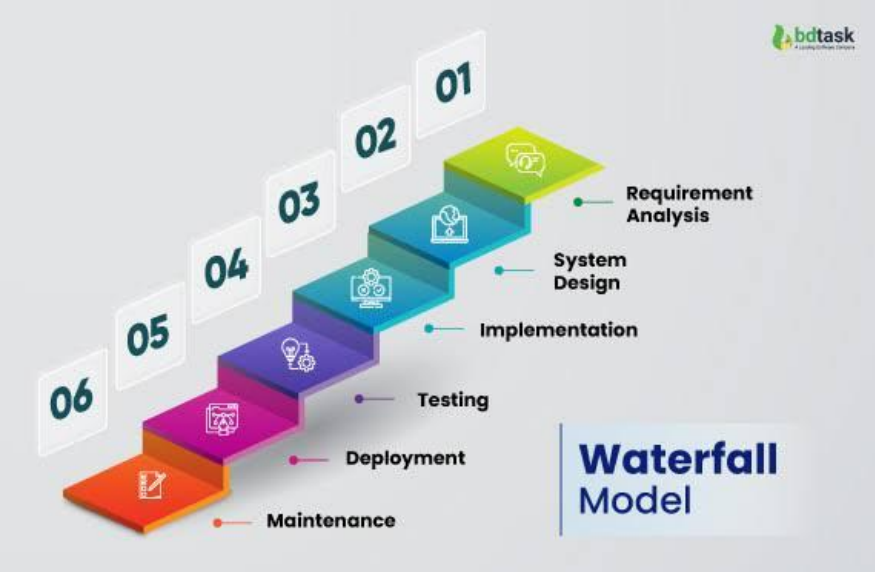

Figure: Waterfall Model

2.2.2. Advantages of Waterfall Model

- **Simple and Easy to Understand:** The Waterfall Model follows a linear and sequential approach, making it easy to implement, even for those new to project management.

- **Easy to Manage:** Due to its rigid structure, the Waterfall Model is simple to manage. Each phase of the project has clear objectives, making it easier to track progress and ensure that the project stays on track.

- **Ideal for Smaller Projects:** The model works well for smaller, less complex projects where the requirements are already well understood, and the scope is unlikely to change during the project lifecycle.

- **Clearly Defined Stages:** The Waterfall Model divides the project into clearly defined stages which make it easier to manage and monitor progress.

- **Process and Results Are Well Documented:** The Waterfall Model emphasizes thorough documentation at each stage, ensuring that the process and results are well-documented. This documentation provides clear records of decisions, design, and progress.

2.2.3. Disadvantages of Waterfall Model

- **Not Suitable for Complex Projects:** The Waterfall Model works best for straightforward projects with well-defined requirements. But for more complex projects especially those involving object-oriented programming or evolving technologies Waterfall's rigid structure can be an obstacle. Object-oriented systems often require flexibility and continuous iteration, which Waterfall doesn't support well, making it less effective for such types of projects.
- **Poor Model for Long and Ongoing Projects:** Waterfall is not well-suited for long term or ongoing projects that evolve over time. For projects that require ongoing updates or extensions, the model can result in inefficient workflows, delays, and difficulty in adapting to new needs.
- **High Amounts of Risk and Uncertainty:** Since the Waterfall Model assumes that all requirements are gathered upfront and changes are difficult to accommodate later, there is a significant amount of risk and uncertainty. If initial assumptions about the project are wrong or the environment changes, it can lead to significant rework, delays, or failure.
- **Delayed Delivery of Working Software:** In the Waterfall Model, no working software is delivered until the later stages, meaning stakeholders won't see a usable product until most of the project is completed. This delay can be problematic for projects that require early feedback or iterative progress.

2.3. Spiral Model

The Spiral Model, also known as the Boehm Spiral Model, is a risk-driven, iterative software development methodology that combines the structured nature of the Waterfall Model with the flexibility of Prototyping. This model is particularly suitable for complex projects with evolving requirements, high uncertainty, or rapidly changing technologies.

2.3.1. Spiral Model Design

Development in the spiral model is structured in a series of iterative loops or spirals. Each spiral cycle consists of four major phases:

1. **Planning:** Define objectives, identify deliverables, and evaluate potential constraints and stakeholder success criteria for the upcoming iteration.
2. **Risk Analysis:** Identify potential technical, financial, and operational risks. Develop mitigation strategies to reduce the likelihood or impact of these risks.
3. **Development:** Build and test a prototype or product increment based on the defined plan and refined requirements. This phase includes coding, unit testing, and incremental integration.
4. **Evaluation:** Gather feedback from stakeholders, evaluate the results of the development phase, and decide on actions for the next cycle—whether to proceed, adjust requirements, or address new risks.

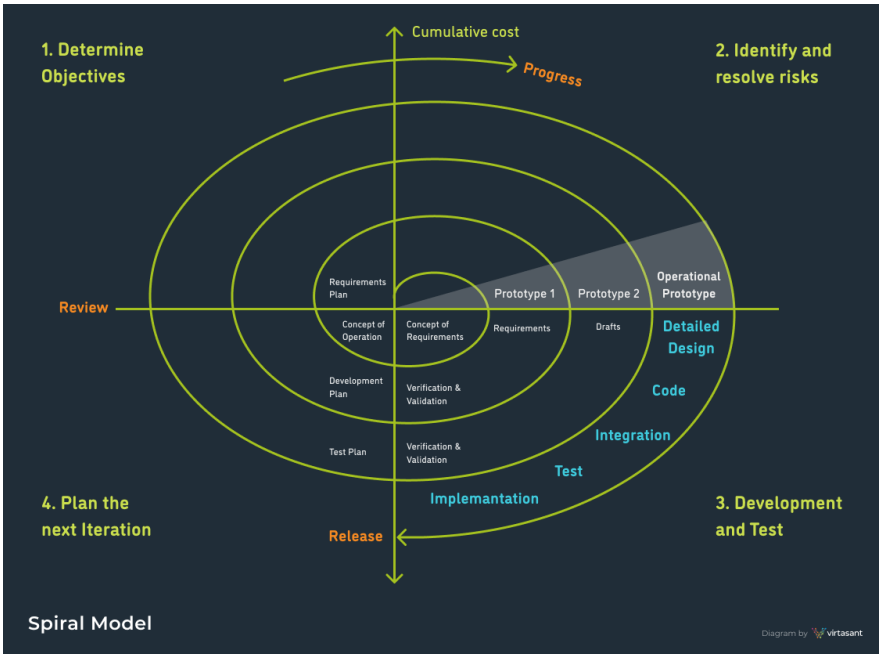

Figure: Spiral Model

Each cycle improves upon the previous one, refining the product through continuous iterations. Notably, the number of cycles often reflects the project's level of complexity and risk.

2.3.2. Spiral Model Terminologies

- **Iteration:** One complete loop through the spiral, from planning to evaluation.
- **Prototype:** A simplified, functional version of the software used for feedback and validation.
- **Risk:** Any uncertain condition that could negatively affect the project.
- **Risk Mitigation:** Strategies implemented to reduce or eliminate potential risks.
- **Baseline:** A stable version of the software produced after a completed iteration.

2.3.3. Spiral Model Procedure

This section describes the working procedure of the spiral model:

1. **Identify Objectives:** Establish project goals, technical requirements, and deliverables.
2. **Analyze Risks:** Evaluate potential risks, such as technological uncertainty or budgetary constraints.
3. **Mitigate Risks:** Develop strategies to address the risks identified, including changes in design or approach.
4. **Development:** Construct a working model or prototype based on the defined strategy.
5. **Evaluation:** Engage stakeholders to review and validate the prototype, collecting feedback for improvements.
6. **Review & Plan Next Iteration:** Based on evaluations, revise plans, and prepare for the next cycle.

2.3.4 Advantages of Spiral Model

- **Flexibility and Adaptability:** Iterative structure allows adjustments to changing requirements and technologies.
- **Risk Management:** Constant focus on identifying and mitigating risks improves the project's chances of success.
- **Stakeholder Feedback:** Regular evaluation loops ensure continuous engagement and alignment with expectations.

- **Improved Quality:** Frequent testing and refinement lead to a more stable and reliable product.
- **Scalability:** Suitable for large-scale, high-budget, or long-duration projects.
- **Early Detection of Flaws:** Prototypes help identify issues in early stages before full-scale development.

2.3.5 Disadvantages of Spiral Model

- **High Complexity:** Managing multiple loops, risks, and feedback channels requires experienced project managers.
- **Unpredictable Costs and Timelines:** Difficulty in accurately estimating budgets and deadlines due to continuous iterations.
- **Documentation Overhead:** Detailed risk analysis and planning require significant time and effort.
- **Requires Skilled Personnel:** Effective risk management and iterative development demand technical expertise.
- **Not Ideal for Small Projects:** The overhead of the spiral model may not justify its use for simple or low-risk systems.

2.4. Agile Model

Agile Methodology is an iterative and incremental approach to software development that emphasizes flexibility, collaboration, and continuous feedback. Unlike traditional models such as Waterfall, Agile allows for dynamic changes and rapid delivery by breaking the development process into short cycles known as iterations or sprints, typically lasting 1 to 4 weeks.

Agile promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, all while encouraging close communication between developers and stakeholders.

2.4.1. Main Characteristics of Agile Methodologies

1. **Iterative and Incremental Development:** Work is divided into small, manageable segments for easier tracking and faster delivery.
2. **Customer Collaboration:** Ongoing involvement of customers or stakeholders ensures that development stays aligned with user needs.
3. **Responsiveness to Change:** Agile embraces changes in requirements even at later stages of development.

4. **Continuous Improvement:** Teams regularly reflect and adapt processes for efficiency and better performance.
5. **Cross-functional Teams:** Agile encourages collaboration among developers, testers, designers, and clients.
6. **Working Software Over Documentation:** Focus is on delivering functional software with minimal reliance on detailed documentation.

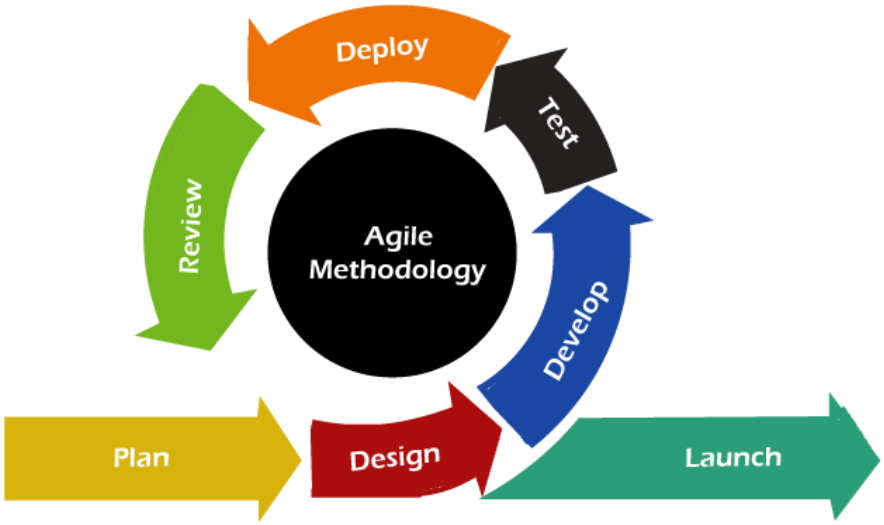

Figure: Agile Methodology

2.4.2. Advantages of Agile Methodologies

- **Higher Product Quality:** Continuous integration, testing, and feedback help detect and fix defects early.
- **Customer Satisfaction:** Frequent delivery and stakeholder involvement ensure better alignment with expectations.
- **Flexibility:** Easily accommodates changes in project requirements and scope.
- **Faster Time to Market:** Short iterations allow for quicker release of functional components.
- **Transparent Process:** Regular updates and meetings enhance communication and visibility.
- **Better Risk Management:** Issues are detected early and corrected quickly.

2.4.3. Disadvantages of Agile Methodologies

- **Scope Creep:** Continuous change requests can lead to uncontrolled expansion of project scope.
- **Limited Documentation:** Lesser emphasis on documentation may cause challenges in maintenance and onboarding.
- **Time-Consuming Meetings:** Frequent discussions (like daily stand-ups and sprint reviews) may disrupt work momentum.
- **High Resource Demand:** Agile requires skilled developers and team members who can work independently and collaboratively.
- **Not Ideal for Large Teams or Fixed Contracts:** Agile may lack predictability in budget and timeline, which can be problematic for large, fixed-scope projects.
- **Dependency on Customer Availability:** Continuous collaboration requires clients to be consistently available, which might not always be feasible.

2.5. Scrum Model

Scrum is an agile framework designed to help teams work together to complete projects in small, manageable pieces over time. It emphasizes collaboration, continuous experimentation, and feedback to improve and deliver value incrementally. Teams using Scrum work in short cycles called sprints, which allow them to refine their processes and adapt quickly to changes. By integrating Scrum, teams can optimize their workflows and deliver high-quality results in a collaborative and efficient manner.

2.5.1 There are some key elements of Scrum

- **Scrum Roles:**
    1. **Scrum Master:** The Scrum Master is accountable for establishing Scrum. They do this by helping everyone understand Scrum theory and practice, both within the Scrum Team and the organization while serving the Scrum Team as well as the larger organization. They helps the scrum team by coaching the team members in self-management and cross-functionality, focus on creating high-value Increments that meet the Definition of Done.

    2. **Product Owner:** A Product Owner is accountable for maximizing the value of the product resulting from the work of the Scrum Team. As a

member of the Scrum Team, the Product Owner provides clarity to the team about a product's vision and goal. All work is derived and prioritized based on the Product Goal in order to deliver value to all stakeholders including those within their organization and all users both inside and out.

3. **Development Team:** Developers are the people on the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint. It is important to remember that a Developer is not necessarily a software developer. They can focus on any type of product work whether software or not and any aspect of helping to design, build, test or ship the product.

- **Scrum Events:**
    1. **Sprint:** Sprints are the heartbeat of Scrum, where ideas are turned into value. The Sprint is the Scrum event that encompasses all of the other Scrum events. They are fixed length periods of work that last one month or less to create consistency and ensure short iterations for feedback in order to inspect and adapt both how work is done and what is being worked on. A new Sprint starts immediately after the conclusion of the previous Sprint.

    2. **Meetings:** Each sprint includes four key sessions. Sprint planning (up to 8 hours), daily scrum (15 minutes), sprint review (around 4 hours), and the sprint retrospective (held post-review). These meetings support effective planning, execution, evaluation, and continuous enhancement of the process.

    3. **Backlog Refinement:** The product backlog must be consistently reviewed and updated to reflect shifting priorities and new requirements. Keeping it up to date is crucial to maintaining its accuracy and ensuring that the team stays focused on the most important tasks.

    4. **Scrum of Scrums:** This approach helps project teams by dividing them into smaller units, with each group selecting a representative to participate in coordination meetings. The primary goal is to manage inter-team dependencies and ensure smooth collaboration across integration points.

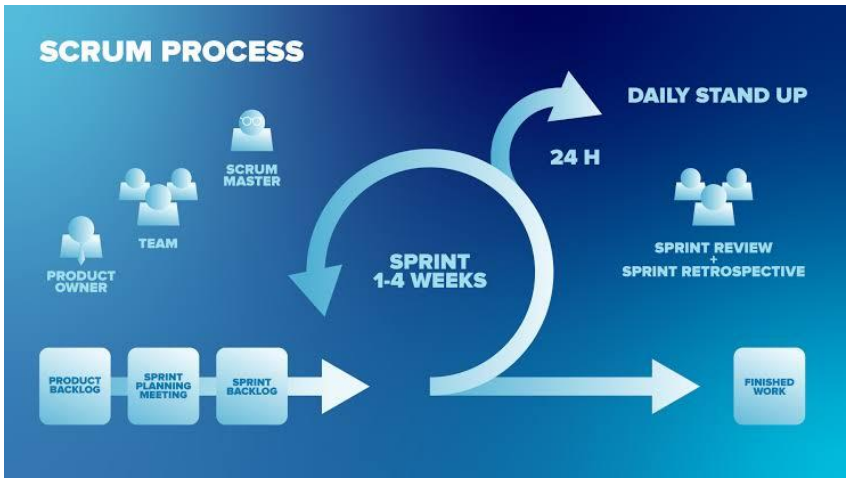

Figure: Scrum Model

2.5.2 Advantages of Scrum Methodology

- **Faster Delivery:** Enables teams to complete project tasks quickly and efficiently.
- **Efficient Resource Use:** Optimizes both time and budget through structured planning.
- **Manageable Workloads:** Breaks large projects into smaller, more manageable sprints.
- **Built-in Testing:** Development and testing occur within each sprint, ensuring quality.
- **Adaptable for Fast Pace:** Ideal for rapidly evolving development environments.
- **Customer-Centric:** Incorporates regular feedback from users and stakeholders.

2.5.3 Disadvantages of Scrum Methodology

- **Scope Creep Risk:** Without a fixed end-date, requirements may expand uncontrollably.
- **Dependency on Commitment:** Success relies heavily on team dedication and collaboration.
- **Scaling Challenges:** Implementing Scrum in large teams can be difficult to manage.

- **Needs Skilled Members:** Requires experienced professionals to execute effectively.
- **Impact of Attrition:** Losing a team member mid-project can severely affect progress.

2.6. Extreme Programming XP Model

Extreme Programming (XP) is a revolutionary agile software development methodology that emerged as a response to the limitations of traditional approaches. Software engineer Ken Beck introduced XP in the 90s with the goal of finding ways to write high-quality software quickly and being able to adapt to customers' changing requirements. At its core, XP places a premium on customer satisfaction and adaptability, offering a dynamic framework that excels in the face of evolving project requirements.

2.6.1. Key characteristics of Extreme Programming

- **Flexibility and Adaptability**: Extreme Programming (XP) is built to handle evolving requirements, even at later stages of development. It prioritizes quick responses to customer input and shifting business priorities.
- **Iterative Development**: XP follows an iterative approach, breaking development into short cycles known as iterations, each delivering a usable segment of the product.
- **Continuous Feedback**: XP relies heavily on constant communication and feedback. With frequent releases and continuous integration, the team can swiftly adapt based on ongoing input.
- **Collaboration**: A strong emphasis is placed on teamwork in XP, encouraging close cooperation among developers, customers, and stakeholders to foster shared responsibility and unified decisions.
- **Test-Driven Development (TDD)**: Writing tests before coding is a fundamental practice in XP, helping to ensure that the code functions correctly and remains stable through changes.
- **Pair Programming**: XP encourages developers to work in pairs—one codes while the other reviews—leading to early error detection and enhanced knowledge transfer.
- **Simple Design**: XP supports maintaining simplicity in design, aiming to keep the code easy to manage and free from unnecessary complications.

- **8.Continuous Integration**: Regular integration of code changes, accompanied by automated builds and testing, helps maintain the consistency and health of the codebase in XP.

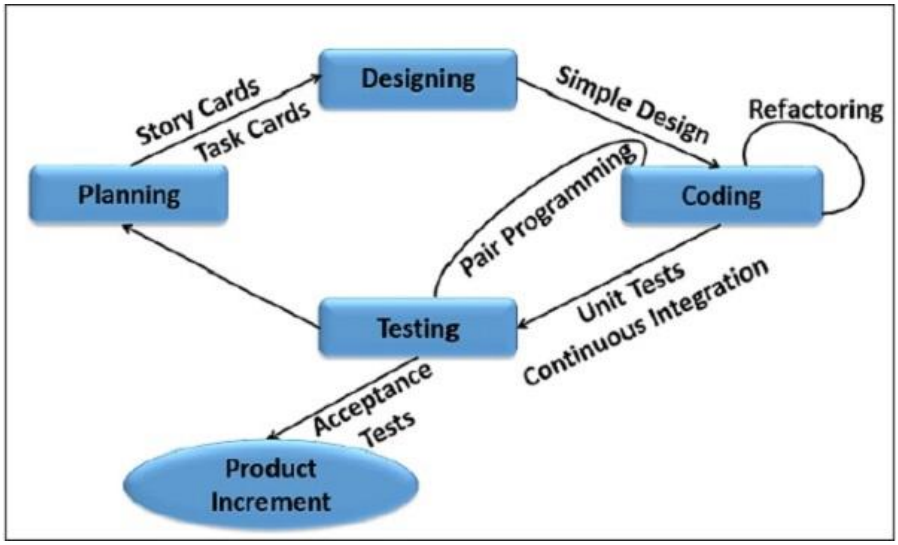

Figure: Extreme Programming Model

2.6.2. Essential Activities of Extreme Programming

- In Extreme Programming (XP), coding is considered the core activity, where developers use high-level programming languages to create executable software.
- Testing holds a crucial place in XP, with test-driven development encouraging developers to write unit tests before the actual implementation to ensure functionality is error-free.
- Listening involves active collaboration between developers and customers, allowing programmers to gather requirements and understand the business context of the software.
- Initially, XP adopts a minimalistic approach to design, focusing on smaller product increments, but as the system matures, greater attention is given to refining the design.

2.6.3. Advantages of Extreme Programming

- Delivers software in short, functional bursts allowing quicker results.
- Frequent releases ensure alignment with stakeholder expectations.
- Bugs are detected early through constant automated testing.
- Collaboration between pairs strengthens code quality.

- The system evolves flexibly with user feedback.
- Reduces documentation overhead by prioritizing code and tests.
- Improves team cohesion and morale through shared responsibility.
- Customer involvement ensures business value is delivered.
- Simpler design leads to easier maintenance and scalability.
- Encourages rapid learning and adaptation for developers.

2.6.4. Disadvantages of Extreme Programming

- High level of collaboration may not suit all team cultures.
- Overemphasis on code can lead to weak documentation.
- Pair programming doubles the developer resource cost.
- Not suitable for large teams or projects requiring rigid structures.
- New team members may struggle without strong mentorship.
- Continuous client involvement may not always be possible.
- Testing overhead can slow down small-scale tasks.
- Code-centric approach may neglect detailed design aspects.
- Hard to manage in distributed teams without strong tooling.
- Stress from constant iterations and delivery expectations can impact team well-being.

2.7. Incremental Model

The Incremental Model in software development involves dividing requirements into standalone modules and progressing through each development stage analysis, design, implementation, testing, and maintenance in iterative steps.

2.7.1. Key characteristics of Incremental Model

1. The entire system is developed as a series of standalone modules.
2. Each module adds new functionality, building upon the previous.
3. Feedback is collected with every increment to refine future ones.
4. Users benefit from early partial versions of the application.
5. Each module goes through the complete SDLC before integration.
6. Ideal for projects with clear, prioritized requirements.
7. System testing is done incrementally with each release.
8. It reduces risks by isolating and testing one component at a time.
9. Allows concurrent development of independent modules.
10. Provides a working system early in the lifecycle.

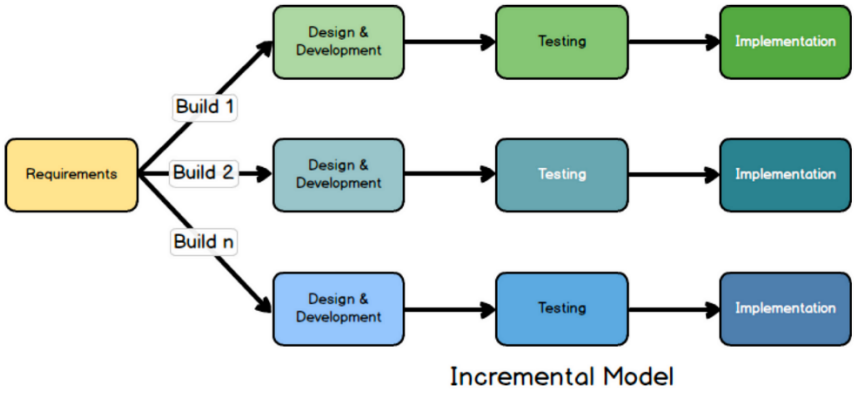

Figure: Incremental Model

2.7.2. Advantages of Incremental Model

- Delivers functional software quickly, supporting early feedback.
- High-priority features can be developed and released first.
- Modules can be tested independently for better quality.
- Budget and resources are easier to allocate incrementally.
- Easier to handle scope changes across iterations.
- Reduces complexity by focusing on small parts at a time.
- Faster reaction to bugs and changes without full rework.
- Offers better risk management as failures are contained.
- Encourages reusable module design.
- Can align deliveries with specific milestones or deadlines.

2.7.3 Disadvantages of Incremental Model

- Integrating modules can become problematic without clear planning.
- Later increments may require redesign of earlier components.
- Scope creep can arise due to ongoing enhancements.
- Dependency between modules can create bottlenecks.
- It requires more rigorous configuration management.
- Total cost may be higher due to repeated testing efforts.
- May not be suitable for systems with tightly coupled components.
- End-user may misinterpret early releases as the final product.
- Requires good architectural planning from the beginning.
- Documentation updates must be made across each increment.

2.8 Benchmark Analysis

This section provides a benchmark comparison of various software development methodologies based on their suitability for the Innovest platform. Key criteria include scalability, adaptability, risk handling, stakeholder engagement, and more.

Criteria	Waterfall	Spiral	Agile	XP	Scrum	Incremental
Scalability	**Low**	**High**	**High**	**High**	**High**	**Medium**
Adaptability	**Low**	**High**	**High**	**High**	**High**	**Medium**
Risk Management	**Low**	**High**	**Medium**	**Medium**	**Medium**	**High**
Customer Interaction	**Low**	**High**	**High**	**High**	**High**	**Low**
Documentation	**High**	**Medium**	**Low**	**Low**	**Medium**	**High**
Expert Requirement	**Medium**	**High**	**High**	**High**	**High**	**Medium**
Cost Efficiency	**Low**	**High**	**Medium**	**Medium**	**Medium**	**Low**
Complexity Handling	**Low**	**High**	**Medium**	**High**	**High**	**Medium**
Speed of Delivery	**Low**	**Medium**	**High**	**High**	**High**	**Medium**
Team Collaboration	**Low**	**Medium**	**High**	**High**	**High**	**Low**

2.9 Selected Methodology for our project

Selecting the right methodology for a project is a pivotal decision that plays a significant role in determining its success. A well-chosen methodology influences the project's efficiency, quality of outcomes, and ability to meet its objectives. When

making this decision, it is essential to assess key factors such as the project's goals, scope, available budget, and the timeline for completion.

After a thorough evaluation of these elements, we carefully selected a methodology that is best suited to achieve the desired results. By aligning the methodology with the project's specific needs, we ensure that resources are utilized optimally, risks are minimized, and milestones are met within the stipulated timeframe. The chosen approach not only complements the project's scale and complexity but also facilitates a structured, well-organized process, contributing to its overall success.

2.9.1 The selected methodology

The selected development methodology for the Innovest platform is the Agile methodology. Agile is a flexible and iterative approach that focuses on continuous improvement, customer feedback, and rapid delivery of functional components. Given the dynamic nature of Innovest which includes real-time updates, investor-founder communication, and evolving user requirements Agile provides the necessary adaptability and user-centric focus.
Agile supports the incremental development of complex systems, making it ideal for projects like Innovest where features need to be deployed progressively and refined based on feedback. The Agile model allows for flexibility, adaptability, and progressive enhancements over time.

2.9.2 Phases SDLC throughout our project

The project followed a simplified SDLC aligned with Agile principles, including:

1. **Planning -** The planning phase involves defining objectives, identifying stakeholders, setting a timeline, and outlining the roadmap to ensure clear direction and resource alignment.
2. **Requirement Gathering** – Identifying core features like real-time updates, direct messaging, and filtering.
3. **Designing** – Designing the platform's user-friendly interface and logical layout.
4. **Development** – Building the core modules iteratively, with regular feature additions.

5. **Testing** – Continuous testing in each sprint to ensure system reliability and performance.
6. **Deployment** – Gradual release of features to users, enabling early feedback.
7. **Maintenance & Feedback Integration** – Addressing bugs and incorporating investor/founder feedback.

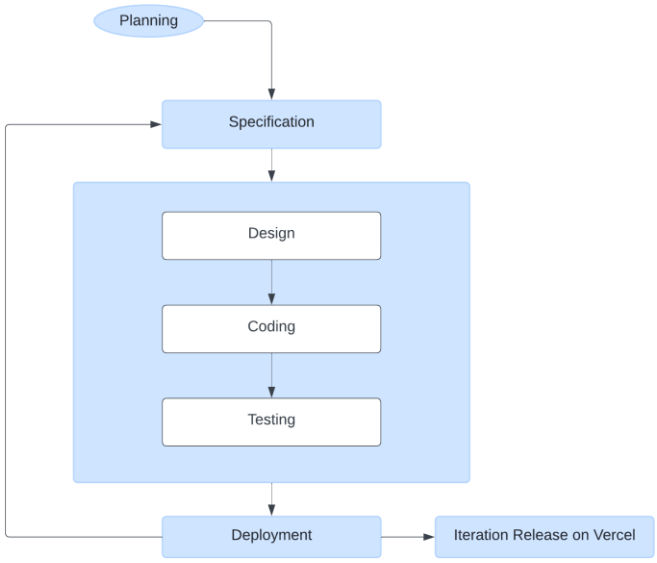

Figure: SDLC of our project

2.10. Choice of methodology

Agile was chosen due to its flexibility, user-centric focus, and iterative approach, which aligns with the evolving nature of startup-investor platforms.

2.10.1 Reasons for choosing the methodology

There were several reasons for choosing Agile as the guiding methodology:

- **Flexibility**: Agile's structure supports changing requirements, which was crucial in a project that involved real-time data and user interaction.
- **Iterative Delivery**: Breaking the project into manageable parts allowed for faster releases and quicker feedback loops.
- **User Feedback Integration:** Real-time communication features required ongoing testing and refinement.

- **User-Centric Design**: Agile encourages continuous feedback from users, helping to shape a platform that meets their actual needs.
- **Improved Team Collaboration**: Agile fosters open communication and collaboration, ensuring that developers, designers, and stakeholders remained aligned throughout the process.

2.10.2 Reasons for not choosing other methodologies

The section describes the reasons for not choosing the other methodologies.

Reason for not choosing Waterfall model:

The Waterfall model follows a linear and sequential approach where each phase must be completed before the next begins. This model is rigid and not ideal for projects where requirements may evolve over time. In the case of Innovest, our platform needed continuous user feedback, frequent design revisions, and incremental development of features like real-time updates and direct messaging. Waterfall lacks the flexibility to accommodate such dynamic changes during development. Additionally, it delays testing and user validation until the end, which could lead to discovering critical issues too late in the process. Therefore, we did not choose the Waterfall model, as it did not align with the agile and user-centric nature of our project.

Reason for not choosing Spiral model:

The Spiral model combines elements of both iterative and waterfall models, with a strong focus on risk assessment and management in each phase. While it offers flexibility and supports iterative development, the Spiral model is often more suitable for large-scale, high-risk projects that require extensive planning and documentation. For Innovest, the overhead of frequent risk analysis and formal documentation in every cycle would have been unnecessary and time consuming. Our project demanded a more streamlined and fast-paced methodology that emphasizes working software over heavy documentation. Hence, we opted against the Spiral model in favor of a lighter and more adaptive Agile approach.

Reason for not choosing Scrum model:

Although Scrum is a popular Agile framework, it introduces rigid structures such as fixed-length sprints, roles (Scrum Master, Product Owner), and formal events like daily standups and sprint reviews. For a compact and flexible team like ours, these additional layers of structure were unnecessary and could slow down development.

We opted for a more generalized Agile approach that allowed us to remain lean while still benefiting from iterative progress and collaboration.

Reason for not choosing Extreme Programming:
Extreme Programming (XP) is highly focused on engineering practices such as pair programming, test-driven development, and continuous integration. While these practices can greatly improve code quality, they demand intensive team collaboration and technical discipline, which may not be practical in a project with limited resources and a focus on rapid prototyping. Since our goal was to develop a functional and user-friendly MVP (Minimum Viable Product) quickly, Agile offered the right balance between flexibility and delivery speed, without the strict demands of XP.

Reason for Not Choosing Incremental Model:
The Incremental Model was not selected for the Innovest project due to its limited flexibility and lack of real-time stakeholder involvement. While it delivers the system in parts, it does not support continuous feedback or dynamic changes once increments are built. The model also requires rigid upfront planning and lacks emphasis on collaboration and rapid iteration—key needs for a fast-moving, user-driven platform like Innovest. Agile methodology offers a more adaptive and interactive development approach, making it better suited to the evolving nature of the project.

Work Breakdown Structure (WBS)

3.1. Introduction to Work Breakdown Structure:
A Work Breakdown Structure (WBS) is a project management tool that systematically breaks down the overall project scope into smaller, more manageable components. It provides a structured view of what needs to be delivered, making it easier to plan, assign responsibilities, and monitor progress. By dividing the project into clear tasks and subtasks, the WBS enhances clarity, improves resource allocation, and supports better control over timelines and deliverables. Essentially, it transforms a complex project into an organized framework, enabling teams to work more efficiently and effectively toward achieving the project goals.

3.2. Principles of Work Breakdown Structure:
The Work Breakdown Structure (WBS) follows a set of key principles that guide how the overall project work is divided and organized. These principles help ensure

the WBS is accurate, complete, and effective in managing project tasks. The major principles are:

1. **100% rule:** The 100% rule is considered the foundation of creating a proper WBS. According to this rule, the total of all the tasks or work packages at a lower level must represent exactly 100% of the work required by their higher-level parent task. It ensures that no part of the work is overlooked and also that no extra, out-of-scope work is included. This rule must be applied consistently across every level of the WBS hierarchy. Adhering to this rule prevents scope creep and helps the team maintain a clear focus on delivering only the required components of the project.

2. **Mutually exclusive elements:** Each task or deliverable in the WBS must be uniquely defined and should not overlap with other components. This means that two tasks should not share responsibilities or duplicate effort. When tasks are mutually exclusive, it becomes easier to assign responsibilities, avoid confusion, and track progress effectively. Clear task boundaries minimize the risk of miscommunication among team members and reduce errors caused by overlapping work.

3. **The 40 Hour Rule of Decomposition:** This rule suggests that when breaking down tasks, each work package should ideally represent about 40 hours of work, which aligns with one full-time work week. If a task is estimated to take more than 40 hours, it should be further divided into smaller, more manageable parts. If it's less than 40 hours, it might be at the right level of detail. This principle helps ensure that no task is too large to manage or too small to track, allowing efficient planning and monitoring.

4. **Outcome oriented:** The WBS should focus on the outcomes or deliverables of the project rather than the actions taken to produce them. It should answer "What needs to be delivered?" rather than "How will it be done?" Deliverables should be named using nouns rather than verbs to emphasize results over processes. This deliverable-based approach helps stakeholders understand the expected project results and facilitates more accurate tracking of progress and completion.

5. **The 4% Rule of Decomposition:** According to this guideline, the smallest tasks (work packages) at the bottom level of the WBS should represent approximately 4% of the total project effort. This provides a useful benchmark

when deciding how deeply to break down tasks. If the task accounts for significantly more than 4%, it may be too broad and need further decomposition. If it's much less, it might be unnecessarily detailed. The 4% rule supports efficiency and balance in the WBS design.

6. **Three levels, Level 2 most important:** A well-structured WBS typically consists of three levels: the top-level representing the full project, the second level showing major deliverables or phases, and the third level detailing specific tasks or work packages. While some branches might need more detailed breakdowns, most parts of the WBS should remain within these three levels. Among them, Level 2 plays a critical role as it outlines the major project components and helps maintain a balance between overview and detail. Keeping most of the structure within these levels ensures manageability without losing sight of the project's overall goals.

3.3. Relevant Rules of Work Breakdown Structure:

The following are the key rules that guide the proper creation and application of a Work Breakdown Structure (WBS) in project management:

1. The WBS should be created in collaboration with the project team.
2. Each level of the WBS must be smaller and more detailed than the level above.
3. Different branches can have different levels of decomposition.
4. Focus should be on outcomes or deliverables, not on actions.
5. Only necessary and essential deliverables should be included.
6. Work Packages should be defined using nouns and further detailed through the WBS dictionary.
7. Levels should be numbered for clarity (e.g., 1, 1.1, 1.2, 2.1, etc.).
8. Each work package should be assigned to a responsible team member or group.

3.4 WBS of Innovest:

This section depicts the Work Breakdown Structure for our project, Innovest:

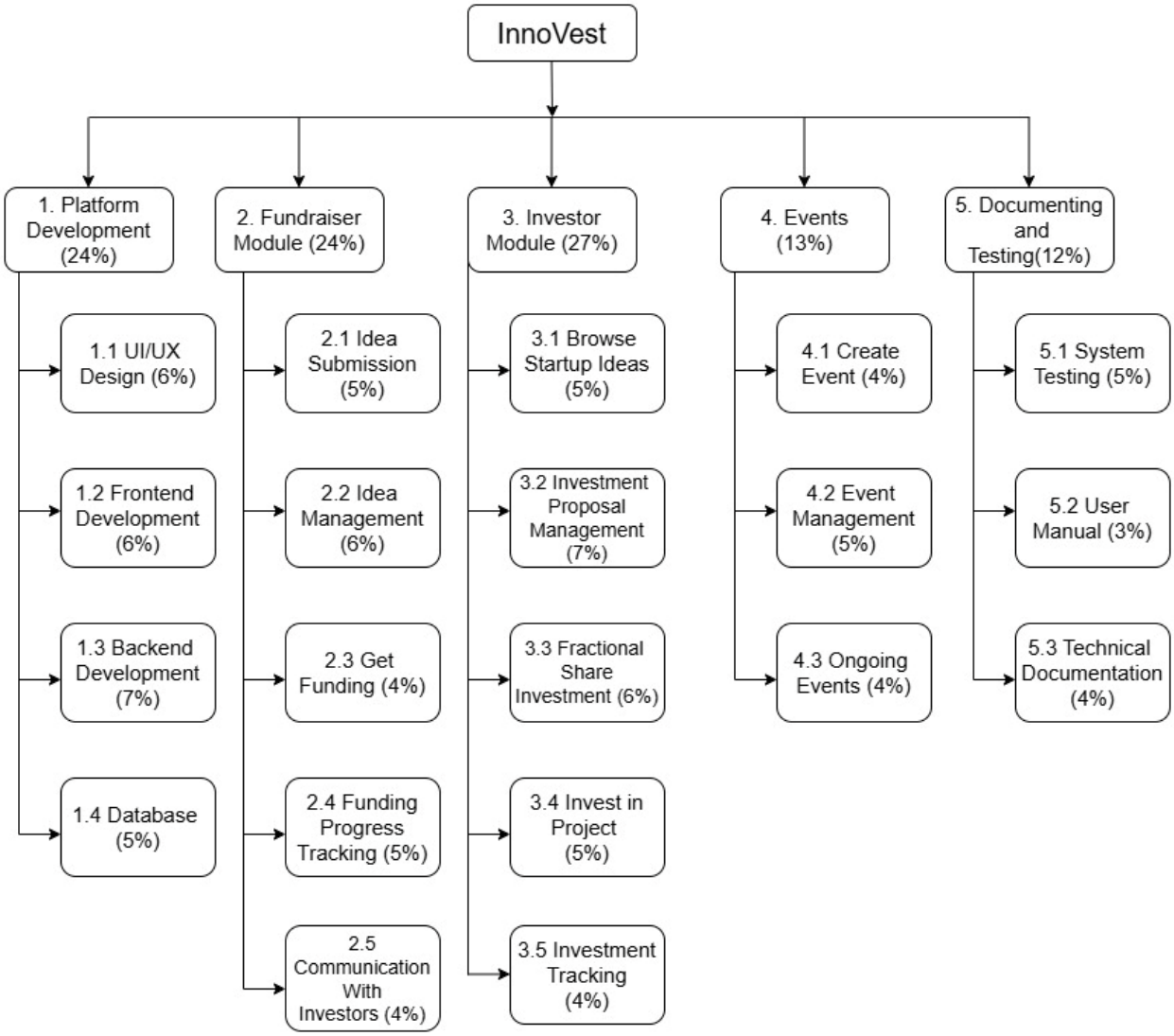

Figure: WBS of Innovest

3.5 Estimated Cost of Innovest:

Task File	Task Owner	Start Date	Due Date	Duration (days)	Estimated Cost (BDT)
UI/UX Design	Ammar	10/04/2025	14/04/2025	4	7000
Front-end Development	Imran	15/04/2025	21/04/2025	6	9000
Back-end Development	Mohin	22/04/2025	29/04/2025	7	10000
Database	Jiku	30/04/2025	04/05/2025	4	6000
Idea Submission	Ananna	05/05/2025	08/05/2025	3	5000

Idea Management	Ammar	09/05/2025	13/05/2025	4	6000
Get Funding	Imran	14/05/2025	17/05/2025	3	4500
Funding Progress Tracking	Mohin	18/05/2025	22/05/2025	4	5000
Communication with Investors	Jiku	23/05/2025	26/05/2025	3	5500
Browse Startup Ideas	Ananna	27/05/2025	30/05/2025	3	5000
Investment Proposal Management	Ammar	31/05/2025	05/06/2025	5	8000
Fractional Share Investment	Imran	06/06/2025	10/06/2025	4	7000
Invest in Project	Mohin	11/06/2025	14/06/2025	3	6000
Investment Tracking	Jiku	15/06/2025	18/06/2025	3	5000
Create Event	Ananna	19/06/2025	22/06/2025	3	4000
Event Management	Ammar	23/06/2025	27/06/2025	4	5000
Ongoing Events	Imran	28/06/2025	01/07/2025	3	4000
System Testing	Mohin	02/07/2025	06/07/2025	4	6500
User Manual	Jiku	07/07/2025	10/07/2025	3	4000
Technical Documentation	Ananna	11/07/2025	14/07/2025	3	5000
Total				61 Days	290,000 BDT

Wideband Delphi

4.1. What is Wideband Delphi:

Wideband Delphi is a consensus-based estimation technique originally developed in the 1940s at RAND Corporation for forecasting purposes. Over time, it has been adapted and proven to be an effective tool for estimation across various industries, particularly in software development projects. It helps teams generate accurate estimates by allowing all members to collaborate and correct each other's assessments, reducing errors and improving accuracy.

The technique addresses a common issue where individuals may not fully understand the scope of what they are estimating. Wideband Delphi also helps uncover key project priorities, assumptions, and deliverables. The process typically follows these steps, as depicted in Figure 4:

Figure 4: Delphi wideband method Wideband

Wideband Delphi Estimation Steps:

1. **Team Selection:** The project manager selects a moderator and an estimation team of 3 to 7 members.
2. **Kickoff Meeting:** The team convenes for the first meeting, where they create a Work Breakdown Structure (WBS) and discuss the project's tasks and requirements.
3. **Individual Preparation:** After the kickoff meeting, each team member individually creates their effort estimate for each task in the WBS.
4. **Estimation Session:** In the second meeting, the team gathers to review and discuss the estimates. They work together to reach a consensus on the estimates.
5. **Assemble Tasks:** After the estimation session, the project manager summarizes the results and reviews them with the team.
6. **Review Results:** The team reviews the final results of the estimation session to ensure accuracy and consensus.

4.2. Wideband Delphi Process for our project:

This section depicts the Wideband Delphi Process of our project Agri-Inn:

4.2.1. Choose the Team:

- The estimation team for the Innovest project consists of 5 members, including one moderator.
- The team members were selected based on their strong background in software development and estimation experience. Each member has a fair understanding of software projects and is capable of making informed estimates.

- We ensured that all members were willing to estimate tasks honestly and collaborate effectively. They also had a good understanding of each other's working styles and maintained a good team dynamic.
- The team has sufficient knowledge of the current startup investment landscape, which is relevant to the Innovest project. Additionally, all team members have worked on software projects in the past, making them qualified to provide educated estimates.
- The team members understand the roles in a project development team, such as managers, developers, designers, architects, QA specialists, analysts, and technical writers.
- Both the moderator and the other members have studied the Delphi Wideband technique and are familiar with its processes.

4.2.2. Kick-off Meeting:

- During the kick-off meeting, the team members were provided with a clear vision and scope of the Innovest project. All team members were already familiar with the project scope from prior discussions.
- The moderator, along with the team members, engaged in brainstorming sessions to generate assumptions and clarify any uncertainties regarding the project.
- A comprehensive task list was created, consisting of 78 distinct major tasks. These tasks covered all the critical aspects required to complete the project successfully.
- An initial set of assumptions was generated, and all team members agreed with these assumptions.
- The team agreed on the task division and ensured that the new tasks and assumptions accurately represented the scope and requirements of the project.

4.2.3. Individual Preparation:

- After the kick-off meeting, the moderator documented the assumptions and tasks generated during the meeting and distributed them among the team members.
- Each team member independently worked on creating their set of estimates for each task, providing their own opinion on the time and effort required.
- During this phase, the team identified any missing tasks in the Work Breakdown Structure (WBS) and added them to ensure that the scope was fully covered. Additionally, some new assumptions were made to refine the overall estimation process.

Individual Preparation Form:

Tasks To Achieve Goal:

Tasks To Achieve Goal	Time (person-days)
UI/UX Design	6
Front-end Development	7
Back-end Development	10
Database	6
Idea Submission	3
Idea Management	4
Get Funding	2
Funding Progress Tracking	3
Communication with Investors	3
Browse Startup Ideas	3
Investment Proposal Management	5
Fractional Share Investment	4
Invest in Project	3
Investment Tracking	2
Create Event	2
Event Management	3
Ongoing Events	2
System Testing	5
User Manual	3
Technical Documentation	4

Calendar Waiting Time, Delays:

Calendar waiting time, delays	Time (person-days)
UI/UX Design	2
Front-end Development	3
Back-end Development	4
Database	2
Idea Management	2
Investment Proposal Management	2

Project Overhead Tasks:

Project Overhead Tasks	Time (person-days)
Family event - Ammar	3

Sickness - Imran	4
University exam pressure - Jiku	5
Tech issue - Mohin	6
Personal emergency - Ananna	4

Assumptions related to individual preparations:

Assumptions
1. Front-end development cannot begin until the UI/UX prototype design is completed and approved.
2. All major features and modules will require a well-structured and clear documentation for successful implementation.
3. Investment and transaction systems (including payment and order processing) depend on the successful implementation of the core investment marketplace.
4. Communication with investors and fund tracking features cannot be developed without a properly structured startup idea management system.
5. Any unexpected system crash or technical failure may delay or halt the development process temporarily.
6. Changes or updates in third-party APIs, frameworks, or libraries (such as Django, MySQL, or payment gateway APIs) might not be compatible with the existing system.
7. Database schema must be finalized before integration begins with the back-end logic to avoid redundant work.
8. User management and access control systems must be in place before allowing any form of communication between investors and startups.
9. All team members will be available according to the schedule, and no long-term absences will impact the project flow.
10. There will be no major changes in project requirements from stakeholders after development starts.

4.2.4. Estimation Session:

- We collected all the estimated forms generated by all team members.
- The estimates were tabulated on a whiteboard, and the totals were plotted on a line for comparison and visualization.
- The estimators discussed the estimates in the group, and through this discussion, we modified and refined the estimations.
- Some disagreements emerged during the process, but we successfully resolved the conflicts by reviewing assumptions and revising the estimates collectively to reach a more moderate and agreed-upon estimation.
- After the group discussion, estimators revised their individual estimates and filled in the "Delta" column on their forms to indicate any changes made during the session.

4.2.5. Assemble Tasks

- The Project Manager gathered all results from the individual preparation phase and estimation session.
- The manager removed redundancies in the estimates and resolved any remaining discrepancies to generate a final task list, which included effort estimates for each task.
- The assumptions made during the estimation process were summarized and added to the final task list for clarity.
- A spreadsheet was created that lists the final estimates from each team member, highlighting the best-case and worst-case scenarios based on the assumptions provided.

4.2.6. Review Result

- Once the results were ready, a final meeting was held to review the estimation results with the team.
- The goal of the meeting was to determine whether the results of the estimation session were sufficient for further planning.
- We discussed whether the estimates made sense and if the estimations were acceptable to proceed with the next steps of the project.
- We ensured that the task list was comprehensive, and the scope of work was almost complete, with no major gaps.

4.3. Individual Estimation Forms

Member: Md. Ammar Hossain

Estimation Form for Delphi Wideband								
Name: Md. Ammar Hossain			**Date:** 10/04/2025			**Estimation Form:**		
Goal statement: To estimate the time to develop the product							**Units:** days	
Category:								
WBS#	**Task Name**	**Estimation**	**Delta 1**	**Delta 2**	**Delta 3**	**Delta 4**	**Total**	**Assumptions**
1	Prototype Development	3	1	-1			3	
2	Database Design and Develop	8	2	1			11	
3	Fundraising System Development	10	2	-2			10	

4	Investing System Development	14	2	3	2		21	Complex analytics involved
5	Event Management Development	11	2	1			14	
6	Admin System Development	9	1	2			12	Role management and data access control
	Delta		10	4	2			
	Total	55	65	69	71			

Member: Md. Al- Emran

Estimation Form for Delphi Wideband								
Name: Md. Al- Emran			**Date:** 10/04/2025		**Estimation Form:**			
Goal statement: To estimate the time to develop the product							**Units:** days	
Category:								
WBS#	**Task Name**	**Estimation**	**Delta 1**	**Delta 2**	**Delta 3**	**Delta 4**	**Total**	**Assumptions**
1	Prototype Development	7	-3				4	
2	Database Design and Develop	14	-3	1			12	Creating backup database server
3	Fundraising System Development	14		-2			12	Secure handling of donor info
4	Investing System Development	10	1				11	
5	Event Management Development	12	0		-3		9	
6	Admin System Development	8	3		0		11	
	Delta		4	1	-3			
	Total	65	69	70	67			

Member: Md. Mohin Khan

Estimation Form for Delphi Wideband								
Name: Md. Mohin Khan			**Date:** 10/04/2025			**Estimation Form:**		
Goal statement: To estimate the time to develop the product							**Units:** days	
Category:								
WBS#	**Task Name**	**Estimation**	**Delta 1**	**Delta 2**	**Delta 3**	**Delta 4**	**Total**	**Assumptions**
1	Prototype Development	12	-2				10	
2	Database Design and Develop	8	1	2			11	
3	Fundraising System Development	15	-2	2	3		18	
4	Investing System Development	8	2	1	-1		10	Complex analytics involved
5	Event Management Development	14	3				17	
6	Admin System Development	7	2	-2	-1		6	Role management and data access control
	Delta		4	3	1			
	Total	64	68	71	72			

Member: Ananna Saha

Estimation Form for Delphi Wideband								
Name: Ananna Saha			**Date:** 10/04/2025			**Estimation Form:**		
Goal statement: To estimate the time to develop the product							**Units:** days	
Category:								
WBS#	**Task Name**	**Estimation**	**Delta 1**	**Delta 2**	**Delta 3**	**Delta 4**	**Total**	**Assumptions**
1	Prototype Development	10	2	-1	1		12	Incomplete design for some features

2	Database Design and Develop	14	-2	3	-1		14	
3	Fundraising System Development	14	1	-2	3		16	
4	Investing System Development	11	2	2	-3		12	
5	Event Management Development	9	3	-3	1		10	
6	Admin System Development	11	3	2	1		17	Role management and data access control
	Delta		9	1	2			
	Total	69	78	79	81			

Member: Md. Shazzad Hossain Jiku

Estimation Form for Delphi Wideband								
Name: Md. Shazzad Hossain Jiku			**Date:** 10/04/2025			**Estimation Form:**		
Goal statement: To estimate the time to develop the product							**Units:** days	
Category:								
WBS#	**Task Name**	**Estimation**	**Delta 1**	**Delta 2**	**Delta 3**	**Delta 4**	**Total**	**Assumptions**
1	Prototype Development	14	3	1	-2		16	Incomplete design for some features
2	Database Design and Develop	9	3	-2	2		12	
3	Fundraising System Development	12	2	1	-1		14	

4	Investing System Development	10	2	-1	3		14	
5	Event Management Development	8	-2	2	2		10	Includes scheduling & notifications
6	Admin System Development	11	-3	2	3		13	
	Delta		5	3	7			
	Total	64	69	72	79			

4.4. Summarized Results of Estimation

WBS #	Task Name	Ammar	Imran	Jiku	Mohin	Ananna	Worst Case	Best Case	Avg (Hi & Lo)	Notes
1	Prototype Development	3	4	16	10	12	16	3	9.5	Slight difference between Ananna and Ammar
2	Database Design and Develop	11	12	12	11	14	14	11	12.5	Estimates are close, low concern
3	Fundraising System Development	10	12	14	18	16	18	10	14	Slightly wide range – discuss implementation complexity
4	Investing System Development	21	11	14	10	12	21	10	15.5	Higher estimates – financial logic and integration concerns
5	Event Management Development	14	9	10	17	10	17	9	13	Uniform but check on platform variability

6	Admin System Development	12	11	13	6	17	17	6	11.5	Small range, can be finalized easily
	Total	71	59	79	72	81	81	59	70	Variations acceptable; discuss higher outliers