Day 1 – Agile

SDLC – Software Development Life Cycle

WWH – What Why How

What is Agile –

Agile Manifesto (4)

Individuals and interactions over processes and tools  
Working software over comprehensive documentation  
Customer collaboration over contract negotiation  
Responding to change over following a plan

12 Principles of Agile

Our highest priority is to satisfy the customer  
through early and continuous delivery  
of valuable software.

Welcome changing requirements, even late in  
development. Agile processes harness change for  
the customer's competitive advantage.

Deliver working software frequently, from a  
couple of weeks to a couple of months, with a  
preference to the shorter timescale.

Business people and developers must work  
together daily throughout the project.

Build projects around motivated individuals.  
Give them the environment and support they need,  
and trust them to get the job done.

The most efficient and effective method of  
conveying information to and within a development  
team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development.  
The sponsors, developers, and users should be able  
to maintain a constant pace indefinitely.

Continuous attention to technical excellence  
and good design enhances agility.

Simplicity--the art of maximizing the amount  
of work not done--is essential.

The best architectures, requirements, and designs  
emerge from self-organizing teams.

At regular intervals, the team reflects on how  
to become more effective, then tunes and adjusts  
its behavior accordingly.

Ceremony -

Sprint – Period (Sprint Planning)

Daily Standup – [What is done, what is pending (the plan for today), any road-blocks]

Retro – [What went well, what went wrong, improvements]

Artifacts of Agile (Documents)

1. Product Backlog
2. Sprint Backlog
3. Burn down/up chart

Epic, User Stories, Task, Sub Task, Estimation, Complexity,

People involved in Agile

1. Product Owner
2. Scrum Master
3. Scrum Team

JIRA, BitBucket, Confluence (Atlassian)

People Flow

Application to monitor smooth movement of people inside office campus. There are different categories of people who enter into office campus. They are Regular Employees (full-time employees), Interns, External Employees (Vendors, Trainers, Support-Staff)

Product Backlog

Sprint Backlog

ChatGPT

**Java Full Stack Web Application with React Development Documentation**

**1. Project Overview**

**1.1 Objective**

Develop a web-based application to manage people flow within an organization, ensuring optimal tracking, monitoring, and analysis of movement.

**1.2 Tech Stack**

* **Frontend**: React, Redux, Material-UI, TypeScript, React Hooks, Axios & Router
* **Backend**: Spring Boot, Spring Security, Hibernate, REST APIs using Spring Boot
* **Database**: PostgreSQL/MySQL
* **CI/CD**: GitHub Actions/Jenkins, Docker, K8S, GCP Cloud Build & Cloud Run
* **Cloud Platform**: Google Cloud Platform (GCP)
* **Authentication**: OAuth 2.0, JWT

**2. Best Practices & Design Patterns**

**2.1 Coding Standards**

* Use meaningful variable and method names
* Follow Java and JavaScript coding conventions
* Write modular and reusable components in React

**2.2 Design Principles**

* SOLID Principles
* DRY (Don't Repeat Yourself)
* KISS (Keep It Simple, Stupid)
* YAGNI (You Ain't Gonna Need It)

**2.3 Design Patterns**

* **Frontend**: Component-Based Architecture, Container-Presentational Pattern
* **Backend**: Repository Pattern, Singleton Pattern, Factory Pattern
* **Database**: Active Record Pattern

**3. CI/CD Pipeline**

**3.1 Version Control & Branching Strategy**

* **Main Branch**: Stable release
* **Develop Branch**: Integration of features
* **Feature Branches**: Individual development branches
* **Hotfix Branches**: Critical fixes

**3.2 CI/CD Implementation**

1. **GitHub Actions or Jenkins** for CI/CD Automation
2. **Docker** for containerization
3. **Kubernetes** for orchestration
4. **GCP Cloud Build** for build and deployment automation
5. **Cloud Run** for scalable service hosting
6. **Monitoring**: Prometheus, Grafana

**4. Agile Project Plan**

**4.1 Timeline & Sprints**

* **Total Duration**: 12 Weeks
* **Sprints**: 12 sprints (5 days per sprint)

**4.2 Product Backlog**

| **ID** | **User Story** | **Priority** |
| --- | --- | --- |
| P1 | As an admin, I want to add, edit, delete users. | High |
| P2 | As a user, I want to check-in and check-out securely. | High |
| P3 | As a manager, I want to view real-time reports of people flow. | Medium |

**4.3 Sprint Backlog & Plan**

**Sprint 1: Project Setup & Environment Configuration**

* Setup GitHub repository and CI/CD pipeline
* Initialize React and Spring Boot projects

**Sprint 2: Authentication & User Management**

* Implement OAuth 2.0 & JWT authentication
* Develop user registration and login API
* Create UI for login and registration

**Sprint 3: Role-Based Access Control**

* Implement user roles (Admin, Manager, Employee[Internal/External])
* Restrict API access based on roles
* Develop frontend components for role management

**Sprint 4: User Check-in/Check-out Feature**

* Develop API for user check-in/check-out
* Implement UI for check-in/check-out
* Store data in PostgreSQL/MySQL

**Sprint 5: Dashboard & Reporting**

* Develop API for reports generation
* Implement real-time dashboard UI
* Integrate with Prometheus/Grafana

**Sprint 6-12: Enhancements, Testing & Deployment**

* Code refactoring and optimization
* Performance testing
* Deployment to GCP Cloud Run
* Final user acceptance testing (UAT)

**5. Conclusion**

This plan ensures a structured and efficient development process while maintaining high coding standards and best practices. Regular sprint reviews will be conducted to refine and improve features.

Perplexity

Building a Java full-stack application with React requires careful planning across technical implementation, agile practices, and DevOps integration. Below is a comprehensive 12-week plan following Scrum methodology with 5-day sprints, incorporating industry best practices from the provided sources.

**Technical Stack & Best Practices**

**Backend (Java):**

* **Spring Boot** for REST API development[1](https://javadzone.com/top-10-java-full-stack-development-best-practices-you-should-follow/)
* **Hibernate** with JPA for database interactions
* **JUnit 5** and Mockito for testing (85%+ coverage target)
* **Spring Security** with OAuth2/JWT for authentication[1](https://javadzone.com/top-10-java-full-stack-development-best-practices-you-should-follow/)

**Frontend (React):**

* **React 22** with Functional Components + Hooks
* **React Router 7** for navigation
* **Material-UI** component library
* **Axios** for API communication[2](https://www.toptal.com/react/react-design-patterns-and-best-practices)

**DevOps (GCP):**

* **Cloud Build** for CI automation[3](https://www.withcoherence.com/articles/ci-cd-gcp-best-practices-for-devops)[4](https://www.pluralsight.com/resources/blog/cloud/cloud-based-ci-cd-on-gcp)
* **Cloud Deploy** with Kubernetes Engine
* **Artifact Registry** for Docker images
* **Cloud Monitoring** for observability

**Product Backlog Structure (Master Backlog)**

| **Epic ID** | **Epic Name** | **Priority** | **User Stories Count** |
| --- | --- | --- | --- |
| E1 | Authentication & Authorization | Critical | 8 |
| E2 | User Profile Management | High | 6 |
| E3 | Department Hierarchy | Medium | 4 |
| E4 | Analytics Dashboard | High | 5 |
| E5 | Notification System | Medium | 3 |

**Sample Sprint Backlog (First 3 Sprints):**

| **Sprint** | **Duration** | **Objectives** | **Key Deliverables** |
| --- | --- | --- | --- |
| 1 | Week 1 | Core setup & Auth | - Spring Boot scaffolding  - JWT implementation[1](https://javadzone.com/top-10-java-full-stack-development-best-practices-you-should-follow/)  - Login UI components[2](https://www.toptal.com/react/react-design-patterns-and-best-practices) |
| 2 | Week 2 | User Management | - CRUD APIs  - User table component  - Input validation |
| 3 | Week 3 | Role System | - Permission service  - Role assignment UI  - API security tests |

**Detailed 12-Week Sprint Plan**

**Phase 1: Foundation (Sprints 1-4)**

1. Environment setup & CI/CD pipeline[3](https://www.withcoherence.com/articles/ci-cd-gcp-best-practices-for-devops)[4](https://www.pluralsight.com/resources/blog/cloud/cloud-based-ci-cd-on-gcp)
2. Authentication system with MFA
3. User management core features
4. Department structure implementation

**Phase 2: Core Features (Sprints 5-8)**  
5. Organization hierarchy visualization  
6. Leave management system  
7. Attendance tracking integration  
8. Basic reporting dashboard

**Phase 3: Enhancements (Sprints 9-12)**  
9. Advanced analytics module  
10. Real-time notifications[2](https://www.toptal.com/react/react-design-patterns-and-best-practices)  
11. Third-party integrations  
12. Performance optimization & security audit[1](https://javadzone.com/top-10-java-full-stack-development-best-practices-you-should-follow/)

**Key Design Patterns**

**Frontend:**

jsx

*// Provider Pattern for State Management [2]*

**export** **const** UserContext = React.createContext();

**function** App() {

**const** [user, setUser] = useState(**null**);

**return** (

<UserContext.Provider value={{ user, setUser }}>

<MainLayout />

</UserContext.Provider>

);

}

**Backend:**

java

*// Builder Pattern for Complex Objects [1]*

**public** **class** UserBuilder {

**private** String email;

**private** Set<Role> roles = **new** HashSet<>();

**public** UserBuilder withEmail(String email) {

**this**.email = email;

**return** **this**;

}

**public** User build() {

**return** **new** User(email, roles);

}

}

**CI/CD Pipeline on GCP**

**Implementation Steps:**

1. **Cloud Build Triggers** on main branch commits[3](https://www.withcoherence.com/articles/ci-cd-gcp-best-practices-for-devops)
2. **Multi-stage Docker builds** for Java/React
3. **Automated Testing:**
   * Unit tests (JUnit/Jest)
   * Integration tests (TestContainers)
   * Security scans (OWASP DC)
4. **Blue-Green Deployment** via Cloud Deploy
5. **Monitoring Integration:**

text

# cloudbuild.yaml excerpt

steps:

- name: 'gcr.io/cloud-builders/mvn'

args: ['clean', 'package']

- name: 'gcr.io/cloud-builders/docker'

args: ['build', '-t', 'gcr.io/$PROJECT\_ID/app:$COMMIT\_SHA', '.']

**User Stories Examples**

**Authentication Epic (E1):**

1. *"As HR, I need MFA login to secure sensitive data"*
   * Acceptance Criteria:  
     ✓ Google Authenticator integration  
     ✓ Session timeout after 15 mins  
     ✓ Failed login attempt tracking
2. *"As employee, I want password reset functionality"*
   * Technical Specs:  
     ✓ Secure token generation  
     ✓ Email validation flow  
     ✓ Rate limiting on reset requests[1](https://javadzone.com/top-10-java-full-stack-development-best-practices-you-should-follow/)

**Risk Mitigation Strategy**

1. **Code Quality:**
   * Daily code reviews
   * SonarQube integration in CI pipeline
   * Pair programming for critical components
2. **Security:**
   * Weekly dependency scans
   * Secrets management via GCP Secret Manager[3](https://www.withcoherence.com/articles/ci-cd-gcp-best-practices-for-devops)
   * RBAC implementation for all APIs
3. **Performance:**
   * Load testing every sprint
   * Query optimization guidelines[1](https://javadzone.com/top-10-java-full-stack-development-best-practices-you-should-follow/)
   * Caching strategy for frequent requests

This plan combines technical excellence from Java/react best practices[1](https://javadzone.com/top-10-java-full-stack-development-best-practices-you-should-follow/)[2](https://www.toptal.com/react/react-design-patterns-and-best-practices) with efficient DevOps automation on GCP[3](https://www.withcoherence.com/articles/ci-cd-gcp-best-practices-for-devops)[4](https://www.pluralsight.com/resources/blog/cloud/cloud-based-ci-cd-on-gcp), while maintaining agile visibility through structured backlogs[5](https://www.atlassian.com/agile/scrum/backlogs)[6](https://www.toptal.com/product-managers/agile/product-backlog-step-by-step-guide). Each sprint delivers shippable increments while building toward the final product vision.

**Citations:**

1. <https://javadzone.com/top-10-java-full-stack-development-best-practices-you-should-follow/>
2. <https://www.toptal.com/react/react-design-patterns-and-best-practices>
3. <https://www.withcoherence.com/articles/ci-cd-gcp-best-practices-for-devops>
4. <https://www.pluralsight.com/resources/blog/cloud/cloud-based-ci-cd-on-gcp>
5. <https://www.atlassian.com/agile/scrum/backlogs>
6. <https://www.toptal.com/product-managers/agile/product-backlog-step-by-step-guide>
7. <https://learn.microsoft.com/en-us/azure/devops/boards/sprints/assign-work-sprint?view=azure-devops>
8. <https://www.shortcut.com/guides/sprint-backlog>
9. <https://www.atlassian.com/agile/project-management/user-stories>
10. <https://www.smartsheet.com/content/user-story-examples>
11. <https://pagepro.co/blog/how-to-build-great-user-stories-examples/>
12. <https://asana.com/resources/sprint-backlog>
13. <https://www.atlassian.com/agile/project-management/sprint-backlog>
14. <https://www.knowledgehut.com/blog/agile/user-stories-examples>
15. <https://www.simpliaxis.com/resources/how-to-create-sprint-backlog>
16. <https://www.syncfusion.com/blogs/post/react-design-patterns>
17. <https://www.expertia.ai/career-tips/10-essential-tips-for-java-full-stack-react-js-developers-43401q>
18. <https://www.freecodecamp.org/news/best-practices-for-react/>
19. <https://www.youtube.com/watch?v=QOnCy_eCDRg>
20. <https://cloud.google.com/build>
21. <https://www.cloudskillsboost.google/course_templates/691>
22. <https://cloud.google.com/kubernetes-engine/docs/concepts/best-practices-continuous-integration-delivery-kubernetes>
23. <https://cloud.google.com/blog/topics/developers-practitioners/devops-and-cicd-google-cloud-explained>
24. <https://productschool.com/blog/product-fundamentals/product-backlog-examples>
25. <https://theproductmanager.com/topics/product-backlog-examples/>
26. <https://www.smartsheet.com/content/product-backlog-templates>
27. <https://people10.com/how-to-create-a-product-backlog-the-right-way/>
28. <https://www.enov8.com/blog/agile-sprint-scheduling-explained-a-detailed-guide/>
29. <https://www.easyagile.com/blog/agile-sprint-planning>
30. <https://www.scrum.org/forum/scrum-forum/76906/8-hour-sprint-planning>
31. <https://www.youtube.com/watch?v=0QbnpNDDa-I>
32. <https://www.parabol.co/blog/user-story-examples/>
33. <https://www.builder.ai/blog/create-app-user-stories>
34. <https://www.esparkinfo.com/blog/react-with-java.html>
35. <https://www.linkedin.com/pulse/best-practices-full-stack-developers-java-spring-boot-jtc-india-fy7ac>
36. <https://www.uxpin.com/studio/blog/react-design-patterns/>
37. <https://fullstackopen.com/en/part1/introduction_to_react>
38. <https://www.youtube.com/watch?v=5PdEmeopJVQ>
39. <https://digital.ai/catalyst-blog/building-cicd-pipeline-gcp/>
40. <https://www.withcoherence.com/articles/google-cloud-ci-cd-best-practices>
41. <https://hevodata.com/learn/gcp-ci-cd/>
42. <https://www.altexsoft.com/blog/product-backlog/>
43. <https://www.shiksha.com/online-courses/articles/explore-product-backlog-basics-blogId-144773>
44. <https://ppm.express/blog/product-backlog/>
45. <https://www.perforce.com/blog/hns/need-inspiring-product-backlog-example-here-are-6>
46. <https://business.adobe.com/blog/basics/sprint-planning>
47. <https://www.6sigma.us/project-management/pi-planning-and-sprint-planning/>
48. <https://www.projectmanager.com/templates/agile-sprint-plan-template>
49. <https://www.easyagile.com/blog/how-to-write-good-user-stories-in-agile-software-development>
50. <https://www.wrike.com/agile-guide/user-stories-guide/>
51. <https://www.mountaingoatsoftware.com/agile/user-stories>
52. <https://getstream.io/blog/user-story-examples/>

Answer from Perplexity: [pplx.ai/share](https://www.perplexity.ai/search/pplx.ai/share)

Copilot

Certainly! Below is a detailed outline for creating a comprehensive documentation for a Java Full Stack application with React, adhering to best practices, design principles, and leveraging GCP for CI/CD. This includes a product backlog, sprint backlog, sprint-wise plan, and user stories. The goal is to deliver a web-based application for managing people flow in an organization within a 12-week timeline.

---

## \*\*Overview\*\*

### \*\*Technology Stack\*\*

- \*\*Frontend\*\*: React.js

- \*\*Backend\*\*: Java (Spring Boot)

- \*\*Database\*\*: PostgreSQL / MySQL

- \*\*Cloud\*\*: Google Cloud Platform (GCP)

- Compute: Cloud Run or Kubernetes

- Storage: Cloud SQL, Cloud Storage

- CI/CD: Cloud Build, Cloud Deployment Manager

- Monitoring: Cloud Monitoring, Logging

- \*\*Best Practices\*\*:

- Code Quality: Use SonarQube for static code analysis

- CI/CD: Automate builds, tests, deployments

- Design Principles: SOLID, DRY, KISS

- Security: OAuth2 authentication, secure APIs, input validation

- Responsive UI and Accessibility Standards

---

## \*\*Product Backlog\*\*

### \*\*Features\*\*

1. \*\*User Management\*\*:

- Employee registration, roles, and permissions

- Authentication (OAuth2)

- Profile management

2. \*\*People Flow Monitoring\*\*:

- Check-in/check-out system

- Visitor management

- Real-time dashboard

3. \*\*Reporting and Analytics\*\*:

- Flow statistics

- Downloadable reports

- Historical data trends

4. \*\*Notification System\*\*:

- Email and push notifications

5. \*\*Admin Panel\*\*:

- Manage users, roles, and permissions

- Audit logs

6. \*\*Integrations\*\*:

- Calendar scheduling

- API for third-party systems

---

## \*\*Sprint-Wise Plan\*\*

Each sprint is 5 days, with time dedicated to planning, implementation, testing, and review.

### \*\*Sprint 1: Project Setup & Initial Framework\*\*

- Tasks:

- Set up project repositories (Git)

- Configure GCP services

- Basic frontend/backend architecture setup

- CI/CD pipeline implementation

- Deliverables:

- Project skeleton with CI/CD automation

### \*\*Sprint 2: User Authentication\*\*

- Tasks:

- Implement OAuth2 authentication

- Role-based access control (RBAC)

- Basic user interface for login

- Deliverables:

- Functional login system

### \*\*Sprint 3: User Management\*\*

- Tasks:

- User registration and profile management APIs

- Frontend screens for user management

- Database schema for users

- Deliverables:

- Complete user management system

### \*\*Sprint 4: People Flow Monitoring (Backend)\*\*

- Tasks:

- APIs for check-in/check-out

- Database schema for flow data

- Backend integration for flow tracking

- Deliverables:

- Functional backend for flow tracking

### \*\*Sprint 5: People Flow Monitoring (Frontend)\*\*

- Tasks:

- Design and implement flow tracking dashboard

- Real-time updates using WebSocket

- Styling and responsiveness

- Deliverables:

- Frontend for flow monitoring

### \*\*Sprint 6: Notification System\*\*

- Tasks:

- Email and push notification APIs

- Notification templates

- Backend integration with events

- Deliverables:

- Basic notification system

### \*\*Sprint 7: Reporting\*\*

- Tasks:

- APIs for generating reports

- Frontend screens for reports

- CSV/PDF download feature

- Deliverables:

- Working reporting module

### \*\*Sprint 8: Admin Panel\*\*

- Tasks:

- APIs for admin operations (audit logs, user role management)

- UI for admin panel

- Database adjustments for logging

- Deliverables:

- Admin panel with key features

### \*\*Sprint 9: Integrations\*\*

- Tasks:

- Implement third-party APIs (e.g., calendar)

- Backend integration for external services

- Deliverables:

- Integrated system

### \*\*Sprint 10: Final Testing\*\*

- Tasks:

- End-to-end testing

- Performance testing

- Security testing

- Deliverables:

- Bug-free, secure application

### \*\*Sprint 11: Deployment\*\*

- Tasks:

- Deployment to production on GCP

- Monitoring setup

- Go-live preparations

- Deliverables:

- Live application

### \*\*Sprint 12: User Feedback & Enhancements\*\*

- Tasks:

- Collect user feedback

- Minor updates based on feedback

- Final documentation updates

- Deliverables:

- Feedback-based enhancements

---

## \*\*User Stories\*\*

Below are sample user stories following the \*\*INVEST\*\* principle:

1. \*\*As a user\*\*, I want to register with my email and set a secure password, so I can access the system.

2. \*\*As an employee\*\*, I want to check in when I arrive and check out when I leave, so my attendance is tracked.

3. \*\*As an admin\*\*, I want to view real-time statistics on people flow, so I can ensure smooth operations.

4. \*\*As an admin\*\*, I want to download daily and monthly flow reports in PDF format, so I can share them with management.

5. \*\*As a visitor\*\*, I want to pre-register online before arriving, so my check-in process is quicker.

---

This outline provides a clear roadmap and structure to execute your project. Let me know if you'd like me to elaborate further or assist with specific aspects! 🚀

Software Required:

* + 1. JDK (21)
    2. Eclipse IDE/STS IDE
    3. Tomcat
    4. Maven
    5. Postman
    6. Docker
    7. Jenkins
    8. K8S
    9. TDD/BDD (Junit/Mockito/Cucumber & Selenium) – Optional

Learning new Foreign Lang (Reference Lang)

1. Learn Characters, Alphabets, Symbols
2. Learn Words
3. Learn Grammer
4. Learn Sentences
5. Learn Essay, Paragraph, Poem etc.,
6. Read/Write/Speak & Understand

Learning New Programming Lang (Ref Lang)

1. Learn Tokens (Characters & Symbols)
2. Learning Keywords (Reserved words)
3. Variables, expressions, statements (syntax)
4. Programs (Set of Instructions)
5. Projects
6. Archival (Jar/War/Ear)
7. Containers (Docker Images) – Docker, K8S
8. Deployment (Automating) – Cloud, CI/CD

Coding, Testing, Reviewing - Development

Infra Provi, Configuring, Deploying – Operations

9944159844 – Siva OS

UML – Unified Modelling Lang -