Chapter 1: Introduction

Project Title and Abstract

Title: Smart Campus Management System with Bus Tracking and Event Notification

Abstract: This project aims to develop an integrated smart campus software system that enables real-time bus tracking and event management for universities. It provides a centralized platform for students to receive bus location updates and personalized event notifications. The solution utilizes GPS, mobile notifications, cloud computing, and microservices architecture to improve campus logistics and student engagement.

Background and Problem Statement

Universities face logistical challenges in managing transportation and ensuring students stay informed about campus events. Students often miss events due to poor communication and experience long waits due to uncertain bus arrivals. There is a need for a real-time, scalable digital solution to streamline these processes.

Objectives of the Project

- Enable real-time GPS tracking of campus shuttle buses.
- Send automated event notifications to students based on location, preferences, and role.
- Provide role-based access for students, event organizers, and administrators.
- Offer dashboards for analytics and system monitoring.
- Ensure secure, scalable, cloud-native architecture.

Scope and Limitations

Scope: - Covers transportation and event modules. - Web and mobile interfaces. - Integration with university authentication system.

Limitations: - GPS reliability depends on hardware. - Notifications depend on mobile connectivity.

Chapter 2: Requirements

Functional Requirements

- FR1: View real-time bus location on map.
- FR2: Student registration/login via SSO.
- FR3: Subscribe to campus event categories.
- FR4: Receive notifications for upcoming events.
- FR5: Event management for organizers.
- FR6: Admin dashboard with analytics.

Non-functional Requirements

- NFR1: System uptime > 99%.
- NFR2: Scalable to 10,000+ users.
- NFR3: Secure login and data encryption.
- NFR4: Usability across devices.

Constraints and Assumptions

- Uses AWS cloud services.
- GPS tracking via mobile/GPS-enabled bus devices.

Chapter 3: Architecture

High-Level Architecture (C4 Model)

- **Context Diagram:** Shows users (students, admins, organizers) interacting with the system.
- **Container Diagram:** Frontend (React, Flutter) > API Gateway > Microservices (Node/Java) > Databases (PostgreSQL, DynamoDB)

Technology Stack

- Frontend: React (web), Flutter (mobile)
- Backend: Node.js or Spring Boot
- DB: PostgreSQL, DynamoDB
- Cloud: AWS (Lambda, API Gateway, S3, SNS, Cognito)

Microservices Architecture

- BusTrackingService
- EventService
- NotificationService
- AuthService
- AdminDashboardService

API Contracts

- RESTful APIs defined with OpenAPI (Swagger)
- Example: POST /events, GET /bus-location

Chapter 4: Detailed Design Using Design Patterns

Class Diagrams and Sequence Diagrams

- Class Diagram: `User`, `Event`, `Bus`, `Location`, `Notification`

• Sequence: Student opens app > Requests bus location > Receives data from BusTrackingService

Design Patterns Applied

• **Observer:** For event notifications.
• **Factory:** For creating notification channels (email, SMS, push).
• **Singleton:** For configuration and logging.
• **Strategy:** For routing logic in notifications.

Chapter 5: Implementation Plan

Module Breakdown

• M1: Authentication
• M2: Bus Tracking
• M3: Event Subscription
• M4: Notification Engine
• M5: Admin Panel

Sprints Mapping (Agile)

• Sprint 1: Auth + Event CRUD
• Sprint 2: Bus tracking + API integration
• Sprint 3: Notifications + Testing
• Sprint 4: UI Polish + Deployment

CI/CD Pipeline

• GitHub Actions or AWS CodePipeline: Build > Test > Deploy

Coding Standards

• ESLint + Prettier (JS), Checkstyle (Java), PEP8 (Python)

Chapter 6: Estimation

Estimation Technique

• Story Points (Agile): T-shirt sizing converted to points

Effort Estimation

• Estimated 300 person-hours

Schedule Estimation

- 4 weeks (4 sprints)
- Sprint 1: Week 1
- Sprint 2: Week 2
- Sprint 3: Week 3
- Sprint 4: Week 4

Chapter 7: Resource Allocation and Project Plan

Team Structure

- Product Owner: 1
- Scrum Master: 1
- Developers: 3
- QA: 1

Resource Matrix

- Developer 1: Auth, Event Module
- Developer 2: Bus Tracking, Notification
- Developer 3: Frontend UI

Tools Used

- GitHub, Jira, Postman, Docker, AWS

Chapter 8: Quality Assurance (QA) Plan

Testing Strategy

- Unit Tests: Jest, JUnit
- Integration: REST API tests with Postman
- System Tests: End-to-end Selenium tests
- Acceptance: User story-based testing

Automation and CI Integration

- Automated test runs in CI pipeline

Security and Performance

- OWASP ZAP for security
- Apache JMeter for load testing

Chapter 9: Deliverables

Software

- Microservices APIs
- Web & Mobile frontends
- AWS deployment scripts

Documentation

- User Manual
- API Docs
- ADRs and Architecture diagrams

Deployment Artifacts

- Docker images
- Helm charts (if Kubernetes used)

Chapter 10: Conclusion and Future Work

Summary

Developed a scalable, real-time smart campus system covering transportation and event engagement using AWS and microservices. Employed best practices in DevOps, architecture, and design patterns.

Lessons Learned

- Importance of event-driven architecture
- Testing early and often improves delivery

Future Enhancements

- AI-based event recommendation
- Real-time heatmaps of bus loads
- Offline event check-in via QR code