PROJECT PLAN – COMP 3004 TEAM 52

Responsibilities:

• **Mohammed Hassan**: code implementation, UML Class Diagram, Testing

• **Fawwaz Ahmed**: code Implementation, Textual Explanation, Traceability Matrix

• **Moaz Sholook**: code Implementation, UML Sequence Diagram, Use Cases

• **Rayan Ordjini**: code Implementation, UML Sequence Diagram, Use Cases

• **Sam Marei**: code Implementation, State Machine Diagrams, Use Cases

Things Left To Do:

• Improve graph implementation

• Enhance implementation of the bolus delivery system

Reason:

We are refining our system based on feedback and working to improve key features to meet project expectations.

Planned Submission Date: April 21st

Week 1 Deliverable - Project Plan

Team Members

- Mohammed Hassan **- 101264727**
- Fawwaz Ahmed **- 101257093**
- Rayan Ourdjini **- 101266109**
- Moaz Sholook - 101260491
- Sam Marei - 101256729

Team Roles & Responsibilities

Primary Roles / Supporting Roles

1. **Project Manager & Coordinator (Mo)**
    - **Coordinate team meetings and sprints**
    - **Track progress against milestones**
    - **Ensure documentation is complete**
    - **Manage GitHub repository**

2. **UI/UX Designer (Rayan)**
    - **Design the user interface matching t:slim X2**
    - **Implement Qt UI components**
    - **Create user-friendly interactions**
    - **Design visual data representations**

3. **Backend Development (Moaz)**
    - **Implement core insulin delivery logic**
    - **Develop Control IQ technology algorithms**
    - **Create data generation and simulation models**
    - **Implement profile management system**

4. **Data Visualization & Storage Developer (Fawwaz)**
    - **Implement data logging functionality**
    - **Create graphing and visualization components**
    - **Develop history tracking systems**
    - **Implement data analysis algorithms**

5. **Testing & Integration Specialist (sam)**
    - **Design and implement test cases**
    - **Ensure quality control**
    - **Create error handling mechanisms**
    - **Develop safety critical components**

Secondary Responsibilities

Each team member will also have secondary responsibilities:

- **: UML Documentation, RTM Creation**
- **: Use Case Development, Video Demo Creation**
- **: Error Handling, State Machine Logic**
- **: GitHub Integration, Code Documentation**
- **: Unit Testing, Implementation Documentation**

Project Timeline & Sprints

Sprint 0: Planning & Setup (March 13 - March 17)

Goal: Project setup, requirements analysis, and design planning

Task	Owner	Due Date	Status
Setup GitHub repository	Mo, Moaz	Mar 13	In Prog
Install and configure Qt on VM	All	Mar 14	Done
Analyze requirements & create use cases	Mo	Mar 15	Done
Create initial class diagram	Fawwaz	Mar 16	Done
Define initial product backlog	All	Mar 17	Done
Submit project plan	Mo	Mar 17	Done

Sprint 1: Core Functionality (March 18 - March 31)

Goal: Implement core insulin pump features and basic UI

Week 1 (March 18 - March 24)

Task	Owner	Due Date	Status
Implement basic UI shell	Jini	Mar 20	Done
Create home screen layout	Jini	Mar 21	Done
Implement user profile model (CRUD)	Moaz	Mar 22	Done
Develop basic insulin delivery logic	Moaz	Mar 23	Done
Create data storage framework	Fawwaz	Mar 23	Done
Implement battery & insulin indicators	Jini	Mar 24	Done
Develop initial test cases	Sam	Mar 24	Done

Week 2 (March 25 - March 31)

Task	Owner	Due Date	Status
Implement manual bolus calculator	Moaz	Mar 26	Done
Create profiles management UI	Jini	Mar 27	Done
Develop basic logging system	Fawwaz	Mar 28	Done
Implement basic error handling	Sam	Mar 29	Done
Integrate components	All	Mar 30	Done
Sprint 1 testing	Sam	Mar 31	Done
Sprint 1 review & demo	All	Mar 31	Done

Sprint 2: Advanced Features (April 1 - April 8)

Goal: Implement advanced features, data visualization, and ensure quality

Week 1 (April 1 - April 4)

Task	Owner	Due Date	Status
Implement Control IQ algorithms	Moaz	Apr 2	Done
Create data visualization graphs	Fawwaz	Apr 2	Done
Develop extended bolus features	Moaz	Apr 3	Done
Implement advanced error handling	Sam	Apr 3	Done
Create detailed history view	Fawwaz	Apr 4	Done
Refine UI components	Jini	Apr 4	Done

Week 2 (April 5 - April 8)

Task	Owner	Due Date	Status
Complete UML documentation	Mo	Apr 5	Done
Create requirements traceability matrix	Mo	Apr 5	Done
Final integration	All	Apr 6	Done
Comprehensive testing	Sam	Apr 7	Done
Record demo video	Mo	Apr 7	Done
Final documentation & submission prep	Mo	Apr 8	Done
Project submission	All	Apr 8	Done

Priority List

High Priority (Sprint 1)

1. Basic UI framework and home screen
2. Profile management system (CRUD)
3. Basic insulin delivery (basal rate)
4. Manual bolus calculator
5. Battery and insulin indicators
6. Basic data storage
7. Fundamental error handling (low battery, low insulin)

Medium Priority (Sprint 2)

1. Control IQ technology algorithms
2. Data visualization and graphing
3. Extended bolus functionality
4. Detailed history view
5. Advanced error handling
6. Automatic suspension at low glucose

Low Priority (If Time Permits)

1. PIN-based lock screen
2. Advanced analytics
3. UI animations and transitions
4. Multiple language support
5. Theme customization

Meeting Schedule

- Sprint Planning: Mondays at beginning of sprint (Mar 13, Apr 1)
- Team Sync: Every Monday, Wednesday, Friday (15 min)
- Sprint Review & Retrospective: End of each sprint (Mar 31, Apr 8)

Communication Tools

- GitHub for code and version control
- Discord for daily communication
- in-person meetings (Depends on Task)

Quality Assurance

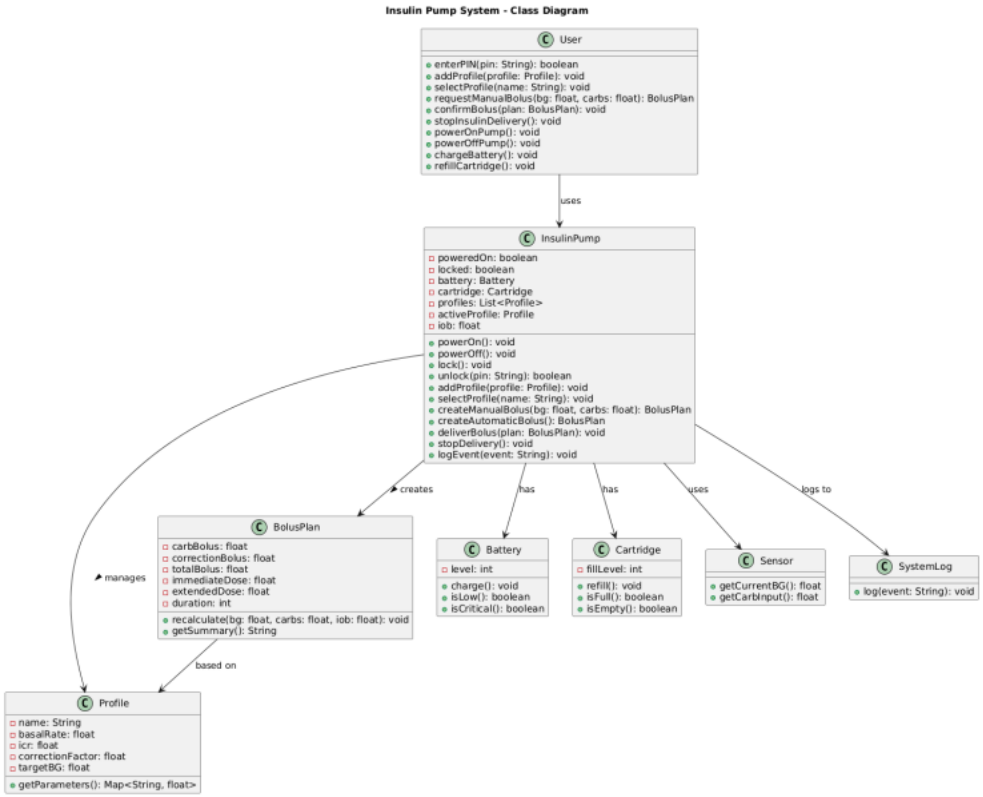
- Unit tests for all critical components
- UI testing for all user interactions
- Safety scenario testing
- Code reviews before merging to develop branch
- Regular builds and integration testing

GitHub Workflow

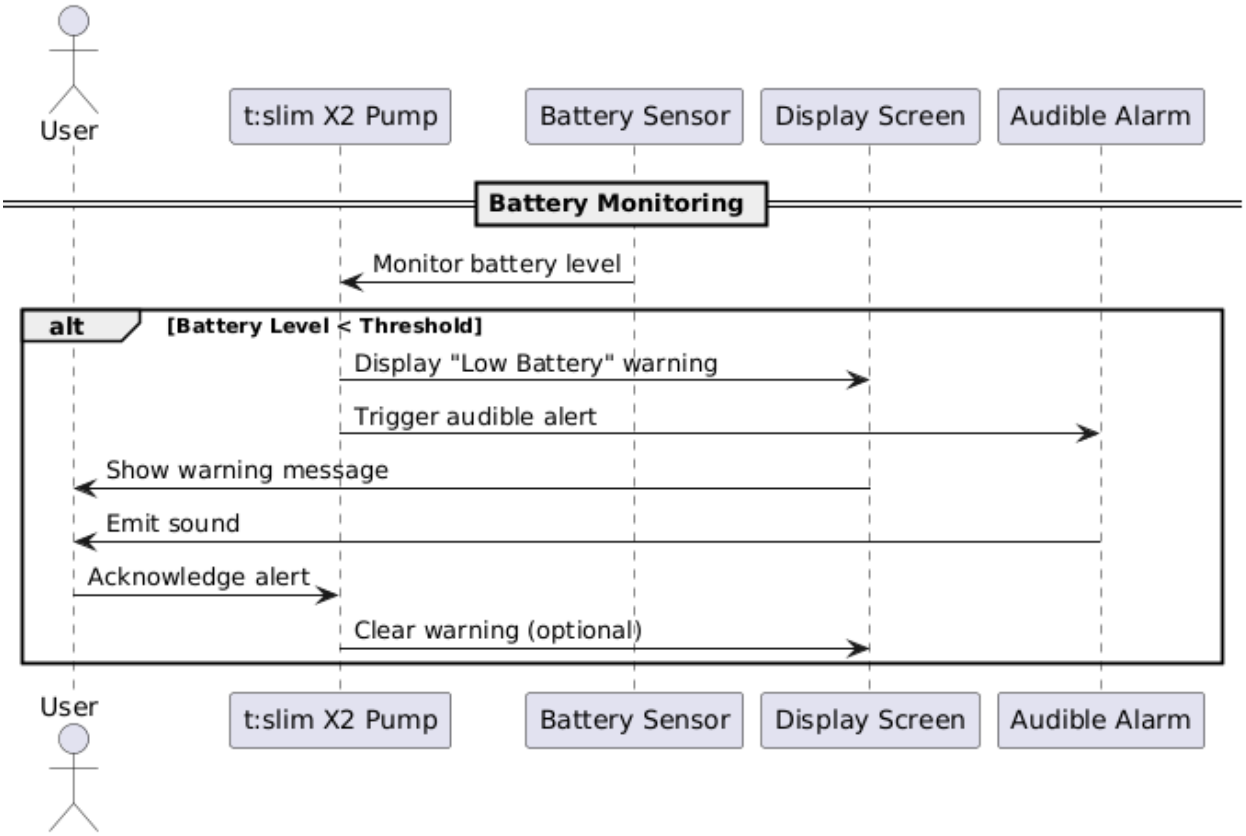
1. main branch for stable releases
2. develop branch for integration
3. Feature branches for individual components
4. Pull request policy: minimum 1 reviewer per PR
5. Regular merges to avoid integration issues

Design Documentation

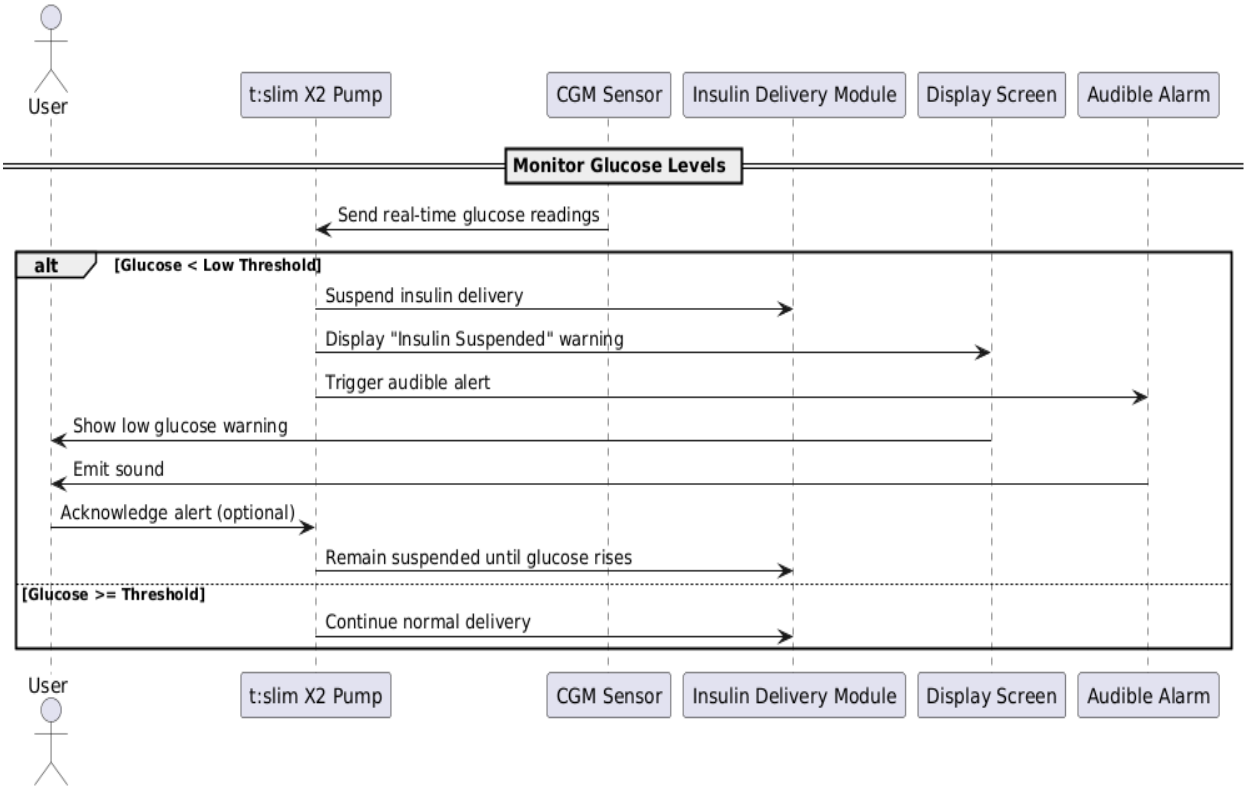
UML CLASS DIAGRAM



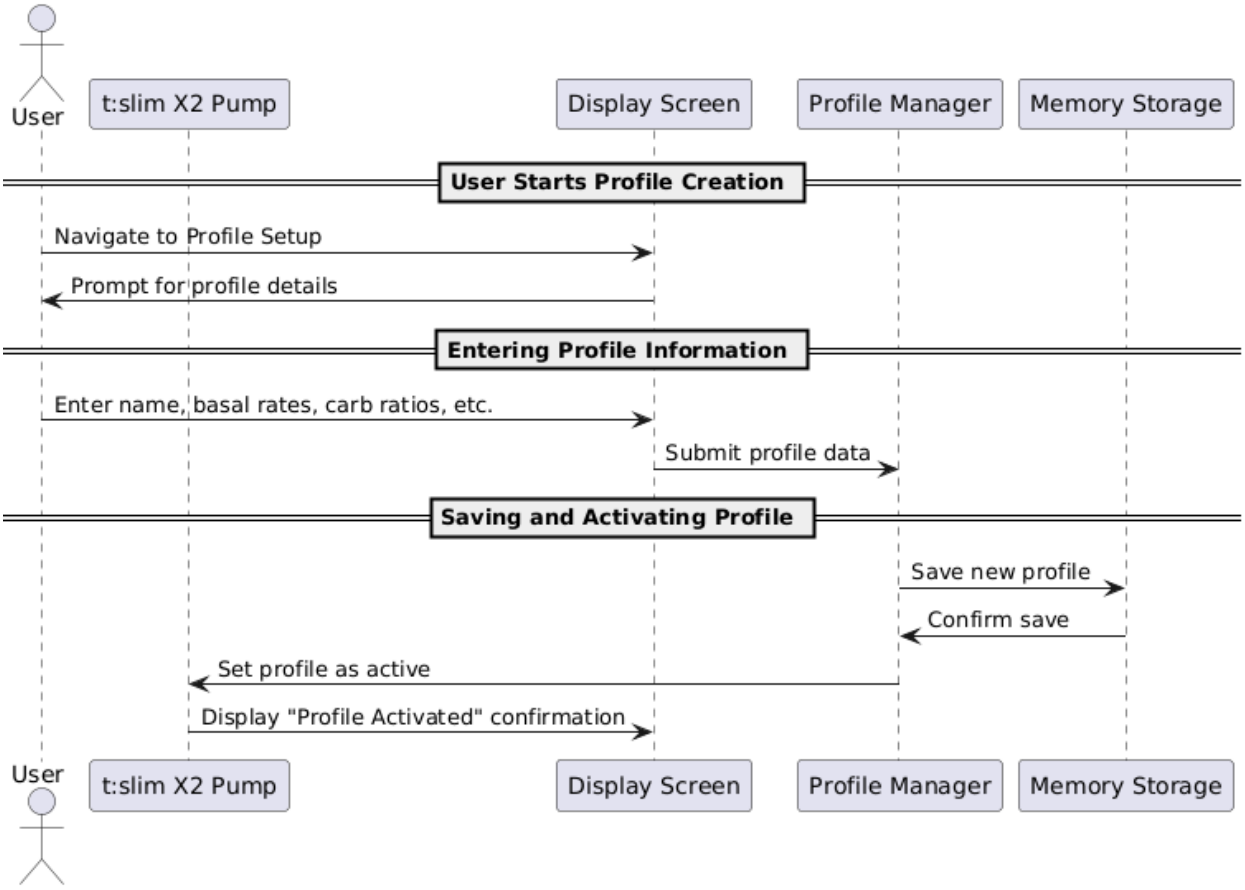
Safety Scenario 1: Low Battery Alert and Response



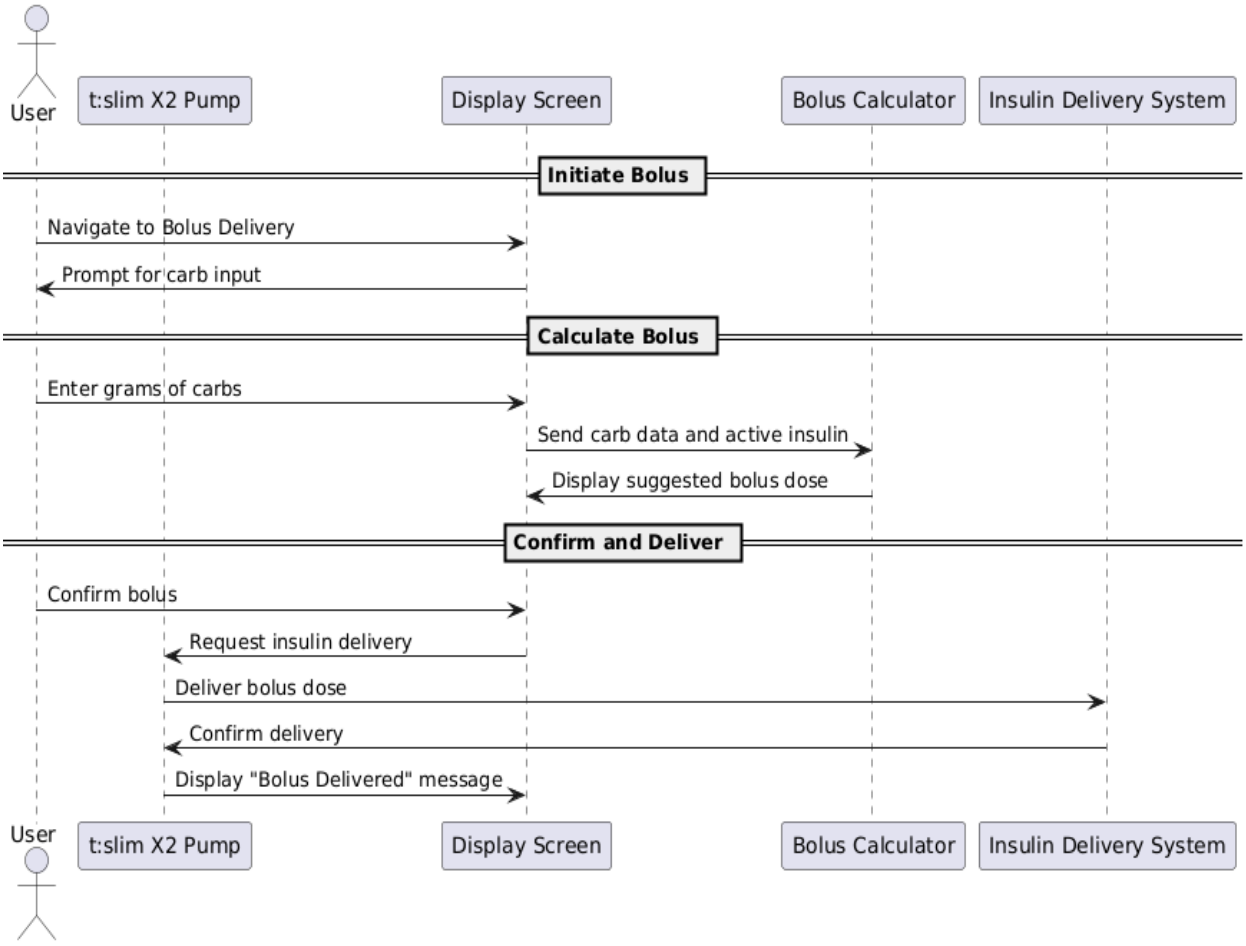
Safety Scenario 2: Automatic Insulin Suspension for Low Glucose



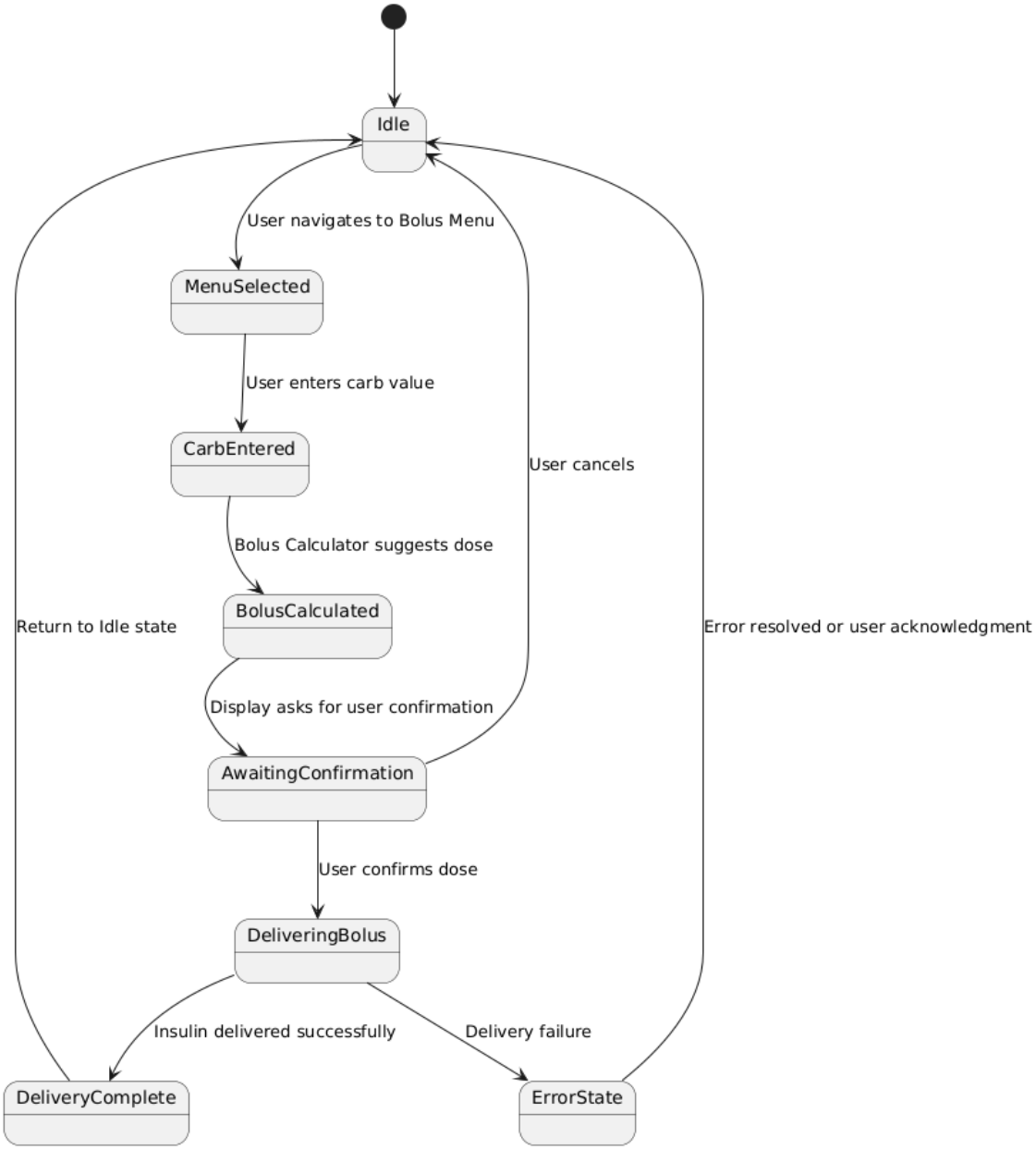
Success Scenario 1: Creating and Activating a Personal Profile



Success Scenario 2: Delivering a Manual Bolus for a Meal



UML State machine diagrams



Textual Explanations:

UML Class Diagram:

The UML Class Diagram illustrates the structural design of the t:X2 insulin pump system, showing all major components and their relationships. This diagram depicts eight key classes including InsulinPump (the central controller), PersonalProfile (storing user-specific insulin settings), BasalRate (defining scheduled insulin delivery rates), BolusCalculator (handling meal and correction doses), SecuritySettings (managing access control), HistoryManager (tracking events), InsulinEvent (representing historical entries), and CGMSystem (interfacing with glucose monitoring). The diagram highlights how these classes interact through associations, compositions, and dependencies, creating a comprehensive view of how data flows through the system and how responsibilities are distributed among components.

Safety Scenario 1: Low Battery Alert and Response

This safety-focused sequence diagram depicts how the t:X2 insulin pump handles potential power issues while maintaining treatment continuity. The diagram shows the PowerManagement system detecting battery levels below 20%, triggering alerts through both visual and audible notifications, and guiding the user to connect the charging cable. It emphasizes that insulin delivery continues uninterrupted during the alert and charging process, highlighting the system's fail-safe design. The sequence concludes with the updating of the battery indicator and logging of the event in the system history, demonstrating how the pump combines user notifications with background safeguards to prevent therapy interruption due to power issues.

Safety Scenario 2: Automatic Insulin Suspension for Low Glucose

This sequence diagram showcases one of the most critical safety features of the t:X2 insulin pump: Control IQ technology's automatic response to dangerously low glucose levels. The diagram illustrates how the CGM system detects glucose falling below 3.9 mmol/L, triggering automatic suspension of insulin delivery without requiring user intervention. It shows the alert notification process, continued glucose monitoring, and the system's automatic resumption of insulin delivery when glucose levels rise above 4.4 mmol/L for a sustained period. The sequence demonstrates the sophisticated automation that protects users from hypoglycemia while ensuring treatment continuity, with both suspension and resumption events properly logged for healthcare provider review.

Success Scenario 1: Creating and Activating a Personal Profile

This sequence diagram traces the interaction flow when a user creates a new insulin delivery profile named "Regular Day." The diagram shows communication between the User, InsulinPump, PersonalProfile, BasalRate, and HistoryManager objects as the user navigates to profile settings, enters customized parameters, and activates the profile. The sequence captures the step-by-step process of entering crucial medical settings including basal rates for different times of day, carbohydrate ratios, correction factors, and target glucose levels. It demonstrates how the system validates these settings, creates the necessary objects, activates the profile, begins insulin delivery according to the new parameters, and logs this critical event in the system history.

Success Scenario 2: Delivering a Manual Bolus for a Meal

The Manual Bolus sequence diagram illustrates the complete workflow for delivering a mealtime insulin dose, a daily critical task for pump users. Beginning with the user accessing the bolus calculator, the diagram shows how the system integrates with the CGM to retrieve the current glucose reading (6.8 mmol/L), accepts the user's carbohydrate input (60 grams), and calculates the appropriate insulin dose (6.0 units) based on the active profile settings. The sequence continues through the confirmation process, actual insulin delivery, updating of the Insulin on Board (IOB) indicator, and logging of the event. This diagram demonstrates the sophisticated interplay between user inputs, automated calculations, and system safety checks during treatment administration.

UML State Machine Diagram

The State Machine diagram provides a dynamic view of the t:X2 insulin pump's operational states and the transitions between them. It captures three primary aspects of the system: power states (Off, On, Sleep), insulin delivery states (Stopped, Basal, Bolus, Suspended), and alert states (Normal, Low/Medium/High Priority). The diagram illustrates how the pump transitions between these states based on user commands (like starting insulin delivery), automated safety responses (such as suspending delivery when glucose is low), system conditions (battery levels), and alert triggers. By showing guards, events, and actions associated with transitions, this diagram reveals the complex decision logic that enables the pump to balance effective treatment with critical safety features, particularly highlighting the automated protective mechanisms of Control IQ technology.

Insulin_Pump_Traceability_Matrix

Use Case	Class	Relevant Methods / Responsibilities	Dependencies / Relationships
Profile Management	ProfilesDialog	createProfile(), editProfile(), deleteProfile(), loadProfile(), selectProfile()	Depends on Profile; interacts with MainWindow
	Profile	getBasalRate(), getICR(), getCorrectionFactor(), getTargetBG(), saveToFile(), loadFromFile()	Used by BolusPlan, ProfilesDialog, MainWindow
	MainWindow	applySelectedProfile(), updateProfileDisplay()	Coordinates with ProfilesDialog
	OptionsDialog	getActiveProfile(), emergencyStop()	May expose profile options or control access
Bolus Delivery	BolusPlan	calculateBolus(), getCorrectionBolus(), getCarbBolus(), getTotalBolus(), updateReservoir()	Uses Profile settings; updates MainWindow
	ManualBolusDialog	inputCarbsBG(), showBolusResult(), confirmBolusDelivery()	Calls BolusPlan; confirms with user
	Profile	getICR(), getCorrectionFactor()	Required for calculations in BolusPlan
	MainWindow	triggerBolusDialog(), showWarning()	Coordinates UI actions; may show alerts if insulin is low
Battery Management	Battery	getLevel(), isCharging(), drain(), charge(), getStatus(), isCritical()	Polled by MainWindow
	MainWindow	updateBatteryStatusUI(), checkBatteryWarnings()	Displays battery level and alerts; calls Battery
Insulin Reservoir Management	MainWindow	updateInsulinLevel(), showInsulinWarning(), preventBolusIfEmpty()	May call BolusPlan or receive updates
	BolusPlan	updateReservoirAfterBolus(), checkInsulinLevel()	Updates reservoir after bolus delivery
Glucose Monitoring	MainWindow	displayGlucoseLevel(), updateGlucoseTrendChart(), showGlucoseWarning()	Uses sensor data; updates UI
	BolusPlan	calculateCorrectionBolus()	Uses glucose levels in calculation
	ManualBolusDialog	inputBG(), previewCorrection()	Passes BG to BolusPlan
System Time Management	MainWindow	displayCurrentTimeDate(), updateClock(), formatTimeDisplay()	Handles real-time updates
	OptionsDialog	setTimePreferences()	Might control date/time formatting options