Z

**Bachelor of Computing System**

**Unitec New Zealand**

5 Nov 2023

Assignment

Cloud Design and development

Table of Contents

[Introduction: 1](#_Toc148452115)

# Introduction:

Managing medication can be a complex task, especially for people who need to take it regularly. This challenge extends to those responsible for their care. In response, we introduce the Everdose Application—a mobile application designed to tackle the intricacies of medication adherence.

Taking medicine isn't always straightforward. For individuals with chronic conditions or specific treatment plans, keeping track of different medications, dosages, and schedules can be confusing. This complexity is even more daunting for caregivers, who want to ensure their loved ones receive the best care possible.

Everdose is like a personal assistant for medicine. It keeps track of when to take each medicine, how much to take, and even reminds you so you don't forget. This means less worry and more confidence in knowing you're getting the right care.

By providing clear schedules, detailed instructions, and timely reminders, Everdose turns the sometimes-complicated task of taking medicine into a routine that's easy to follow. This not only reduces stress but also helps the treatment work better.

# Vision Statement

# Product Backlog

The product blog consists of user stories from patient and caregiver, the user stories which achieves the essential functionality of the app is considered high as well as dependency.

## User stories:

|  |  |  |
| --- | --- | --- |
| ID | Description | Priority |
|  | As a patient,  I want to be able to register an account with login details,  So that only registered authorized users can see or update my medication schedules. | High |
|  | As a patient,  I want to be able to set up a schedule for my prescribed medication,  So that I can keep track of all my medications. | High |
|  | As a patient,  I want to be able to view my medication history in the app,  So that I can see whether I've missed any doses. | High |
|  | As a patient,  I want to be able to receive reminders for when I need to take medications,  So that I don't forget to take them when I need them. | High |
|  | As a patient,  I want to be able to view my medication history in the app,  So that I can see whether I've been taking my medication on time. | Medium |
|  | As a caregiver,  I want to receive notifications when the patient misses a medication dose,  So that I can remind them to take it. | High |
|  | As a caregiver responsible for multiple patients,  I want to be able to manage medication schedules for multiple patients within the app,  So that I can efficiently monitor and provide care for each patient. | High |
|  |  |  |

## Requirements and Specifications

# Prototype and Scope

# Application Architecture

This architectural design outlines a medication tracking application using Amazon Web Services (AWS). It employs AWS services for user management, data storage, notifications, security, and scalability. The design emphasizes secure patient management, caregiver support, medication reminders, and data analytics, ensuring a robust and flexible solution for healthcare needs.

**1**. User Interface (UI):

* Develop the user interface as a web application.
* Host the UI using Amazon S3 for static web content or AWS Elastic Beanstalk for web applications.

2. Authentication and Authorization:

* Use Amazon Cognito for user authentication, including OAuth 2.0 and JWT support.
* Implement fine-grained access control with AWS Identity and Access Management (IAM).

3. Patient Management:

* Store patient profiles and caregiver access permissions in Amazon DynamoDB, a NoSQL database for flexibility and scalability.

4. Medication Schedule Management:

* Store medication schedules in DynamoDB.
* Create a RESTful API using AWS API Gateway and AWS Lambda for interaction with schedules.

5. Medication Reminder Service:

* Use AWS Simple Notification Service (SNS) for sending reminder notifications.
* Create a scheduled AWS Lambda function to send reminders.

6. Caregiver Management:

* Store caregiver profiles and relationships in DynamoDB.
* Implement RESTful API for caregiver management using API Gateway and Lambda.

7. Data Storage:

* Use Amazon RDS (Relational Database Service) for storing structured data, such as user profiles, caregiver information, and historical data.
* Use Amazon DynamoDB for unstructured data and high-velocity data storage.

8. Patient-Caregiver Communication:

* Use AWS Simple Notification Service (SNS) for sending notifications to caregivers.
* Configure email and SMS notifications using Amazon SES and Amazon SNS, respectively.

9. License Agreement Management:

* Store license agreement acceptance status in DynamoDB.
* Implement logic to display the agreement in the UI.

10. Security Layer:

* Leverage AWS Web Application Firewall (WAF) to protect against web application attacks.
* Use AWS Certificate Manager for SSL/TLS certificates.
* Secure APIs using API Gateway with built-in security features.

13. Web Clients:

* Use AWS Mobile SDK for seamless integration with AWS services on mobile platforms.

14. High Availability and Load Balancing:

* Implement AWS Elastic Load Balancing (ELB) to distribute traffic and ensure high availability.

15. Backup and Disaster Recovery:

* Implement regular backups using AWS Backup for critical data.
* Set up disaster recovery using AWS Disaster Recovery services.

16. Monitoring and Logging:

* Use Amazon CloudWatch for application and infrastructure monitoring.
* Configure AWS CloudTrail for audit and log trail.

# User Interfaces