A logo of a university

AI-generated content may be incorrect.

Cape Town Festival - Mini Project Submission Agenda

**INF4027W: System Development Project II**

Developer: Mulalo Rasivhaga, RSVMUL002

**Table of Contents**

[1. System Overview 3](#_Toc191656756)

[2. User Roles Implementation 4](#_Toc191656757)

[3. Event Management Features 5](#_Toc191656758)

[4. Dashboard & Reporting 6](#_Toc191656759)

[5. Additional Features Implemented 6](#_Toc191656760)

[6. Challenges and Solutions 7](#_Toc191656761)

[8. Future Enhancements 8](#_Toc191656762)

[9. Scalability considerations 9](#_Toc191656763)

# 1. System Overview

The CT Festival is a vibrant celebration of culture, arts, and community engagement. The festival aims to bring together diverse groups of people to enjoy various events, including music performances, art exhibitions, food stalls, and workshops. The application facilitates user engagement by allowing attendees to RSVP for events, provide feedback, and access event information seamlessly.

**System Architecture and Technologies**

The system is built using the Flutter framework for cross-platform mobile application development. It leverages Firebase for backend services, including authentication, Firestone for real-time database management, and Cloud Functions for server-side logic. The architecture follows a client-server model, where the mobile app interacts with Firebase services to manage user data and event information.

**Database structure**

The database is structured using Firestore, which organizes data into collections and documents. Key collections include:

* **Users**: Stores email first and last name, gender and age.
* **Admins**: Sotes email, first name and last name
* **Events**: Contains event details such as title, description, date/time, location, category, image URL, latitude, longitude, maximum participants, start and end date.
* **RSVPs**: Tracks user RSVPs for events, including attendance status and user ID.
* **Ratings**: Stores user ratings, comment, RSVP ID and User ID.
* **Archived Events**: Transfers events to a separate collection with the same field details

A screenshot of a computer

AI-generated content may be incorrect.

# 2. User Roles Implementation

**Administrator Role Functionality**

Administrators have access to a dedicated dashboard where they can manage events, view analytics, and view user feedback. They can create, edit, and archive events, as well as monitor attendance and ratings.

Administrators are assigned based on their email address. All users who register with a ‘myuct.ac.za’ or ‘uct.ac.za’ email address are automatically assigned to have administrator privileges.

**Visitor/Attendee Role Functionality**

Visitors can browse events, RSVP, provide feedback, and view their RSVP history. They can also register for an account using their email with verification features to ensure email addresses used are authentic.

# 3. Event Management Features

**Event Creation and Management (Admin)**

Administrators can create and manage events through the admin dashboard. They can input all necessary details and update them as needed.

**Event Properties Implementation**

Events include the following properties:

Title: The name of the event.

Description: A brief overview of what the event entails.

Date/Time: The scheduled date and time for the event.

Max Participants: The maximum number of attendees allowed.

Category: The type of event (e.g., music, art, workshop).

Location: The venue where the event will take place.

**RSVP System and Attendance Tracking**

The application allows users to RSVP for events, which updates their attendance status in the database. Administrators can track attendance metrics for each event on the dedicated RSVP analytics Dashboard.

**Rating and Feedback Collection**

Users can rate events on a 5-star scale and leave comments. This feedback is stored in the database and can be analysed by administrators. On the home screen comments regarding the specific event are also visible for the public through the event preview dialog.

# 4. Dashboard & Reporting

**Festival Dashboard implementation**

The festival dashboard provides administrators with an overview of event performance, user engagement, and feedback metrics.

**Visitor demographics**

1. Total Number of Users
2. Gender Distribution across platform
3. User Table
4. Age Distribution across platform

Event-related statistics

1. Total Number of Events
2. Total Number of RSVPs
3. RSVP leaderboard
4. RSVP distribution per event (Pie chart)
5. RSVP distribution per category
6. RSVP Details per event (Table)

**Comment display and management**

Administrators can view comments left by users on RSVP analytics page. The home event preview dialogs cards allow for all users to view recent comments for the specific event.

# 5. Additional Features Implemented

**Google Maps integration for event locations**

The application integrates Google Maps to display event locations, making it easier for users to find venues. Its implementation can be viewed on the event preview dialog card.

**Weather API integration for event forecasts**

Weather forecasts are provided for event dates and the specific location of the event, helping attendees plan their visits accordingly. Its implementation can be viewed on the event preview dialog card.

**Email notifications**

Newly registered users are emailed to verify their email address.

**Deployment details**

The application is deployed on Firebase Hosting, ensuring fast and reliable access for users. The Link for deployment is as follows:

Project Console: <https://console.firebase.google.com/project/ct-festival/overview>

Hosting URL: <https://ct-festival.web.app>

# 6. Challenges and Solutions

**Technical challenges encountered**

1. State Management

Challenge: Managing the state of the application effectively was a significant challenge, especially with multiple components needing to access and update shared data (e.g., user RSVPs, event details). As the application grew, it became increasingly complex to ensure that the UI reflected the current state of the data without unnecessary rebuilds or performance issues.

How Problems Were Addressed:

Use of Riverpod: The project implemented Riverpod as a state management solution. Riverpod provides a more robust and flexible way to manage state compared to traditional methods. It allows for better separation of concerns, making it easier to manage state across different parts of the application.

Scoped Providers: By using scoped providers, the application could limit the rebuilds to only the widgets that depend on the specific state, improving performance and responsiveness.

Future Builders: For asynchronous data fetching (like RSVPs and event details), Future Builders were used to manage loading states and errors effectively, ensuring a smooth user experience.

**2. Responsive Design**

Challenge: Creating a responsive design that works seamlessly across various screen sizes and orientations was another challenge. The application needed to provide a consistent user experience on both small mobile devices and larger tablets.

How Problems Were Addressed:

Flexible Layouts: The use of Flutter's layout widgets, such as Flexible, Expanded, and MediaQuery, allowed for dynamic adjustments based on screen size. This ensured that UI elements resized and repositioned appropriately.

Responsive Breakpoints: Implemented breakpoints to adjust the layout and styling based on the device's screen size. This included changing font sizes, padding, and the arrangement of UI components to maintain usability and aesthetics.

Testing on Multiple Devices: The application was tested on various devices and screen sizes to identify and fix layout issues. This iterative testing helped ensure that the design was truly responsive and user-friendly.

# 8. Future Enhancements

**Features planned for future iterations**

Enhanced Analytics: More detailed analytics features for administrators to gain deeper insights into user behaviour.

Social Media Integration: Allow users to share events on social media platforms.

Event Reminders: Push notifications for upcoming events based on user preferences.

# 9. Scalability considerations

The application is designed to scale with increased user load by utilizing Firebase's serverless architecture, which can handle spikes in traffic without performance degradation.