||JAI SRI GURUDEV||

ADICHUNCHANAGIRI UNIVERSITY

Faculty of Engineering, Management & Technology

B G S INSTITUTE OF TECHNOLOGY

B G NAGARA, KARNATAKA-571448

A Mini Project Report
on

"EVENT MANAGEMENT SYSTEM"

Submitted in partial fulfillment for the academic year 2023-2024

Bachelor of Engineering

in

Information Science and Engineering

Submitted by

LAASYA M S	21ISE027
KEERTHANA RAJ	21ISE025

Under the Guidance of:
Mr. Kiran Kumar D
Asst. Professor,
Dept. of ISE
BGSIT, BG Nagara

DEPARTMENT OF INFORMATIONSCIENCE AND ENGINEERING

B G S INSTITUTE OF TECHNOLOGY

B G NAGARA-571448
2023-2024

||JAI SRI GURUDEV||

ADICHUNCHANAGIRI UNIVERSITY

Faculty of Engineering, Management & Technology

B G S INSTITUTE OF TECHNOLOGY

B G NAGARA, KARNATAKA-571448

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project entitled **"EVENT MANAGEMNET SYSTEM"** is a bonafide work carried out by **Ms. LAASYA M S**, bearing **USN:21ISE027** and **Ms. KEERTHANA RAJ**, bearing **USN:21ISE025** a bonafide students of **B G S Institute of Technology**, **B G Nagara** in partial fulfillment for the award of **Bachelor of Engineering in Information Science and Engineering of Adichunchanagiri University, B G Nagara** during the academic year 2023-2024. Itis certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library.

Signature of Guide:	**Signature of HOD:**
------------------------------	------------------------------
Mr. Kiran Kumar D	**Dr. Siddartha B K**
Asst. Professor,	Assoc. Professor and Head
Dept. of ISE	Dept. of ISE
BGSIT, BG Nagara	BGSIT, BG Nagara

External Viva

Name of the Examiners:	**Signature with Date:**
1. ____________________	____________________
2. ____________________	____________________

ACKNOWLEDGEMENT

While presenting this Mad Mini Project on **"EVENT MANAGEMENT SYSTEM"**, We feel that it is our duty to acknowledge the help rendered to us by various person. Firstly, we thank God for Showering blessings on us. We are grateful to our institution BGS Institute of Technology for providing us a congenial atmosphere to carry out the project successfully.

We would like to express our heartfelt gratitude to **Dr. B N Shobha, Principal BGS Institute of Technology** for extending his support.

We would also like to express our heartfelt gratitude to **Dr. Siddartha B K, Associate Prof, & HOD Department of Information Science and Engineering** whose guidance and support was truly invaluable.

We are very grateful to our guide, **Mr. Kiran Kumar D, Assistant Professor, Department of ISE, BGSIT, B G Nagara,** for his able guidance and valuable advice at every stage of project which helped us in successful completion of our project.

We would also have indebted to our Parents and Friends for their continued moral and material support throughout the project and helping us in finalize the presentation.

LAASYA M S	**(21ISE027)**
KEERTHANA RAJ	**(21ISE025)**

ABSTRACT

In today's dynamic and interconnected world, organizing events efficiently and effectively requires sophisticated tools and technologies. The Event Management App presented in this abstract aims to revolutionize the way events are planned, managed, and experienced. This innovative application integrates cutting-edge features such as Virtual Reality (VR) and Augmented Reality (AR) for immersive venue visualization and customization. Leveraging Artificial Intelligence (AI), the app offers intelligent recommendations for venues, vendors, and event themes based on user preferences and historical data. Real-time collaboration capabilities facilitate seamless communication and coordination among event organizers, vendors, and participants, enhancing productivity and reducing logistical challenges. By integrating these comprehensive functionalities into a user-friendly interface, the Event Management App aims to elevate the event planning and execution process, ensuring memorable and successful events while meeting the diverse needs of organizers and attendees alike in today’s digital age.

CONTENTS

CHAPTER 1
INTRODUCTION

Mobile application development is the process to making software for smartphones and digital assistants, most commonly for Android and iOS. The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser. The programming and markup languages used for this kind of software development include Java, Swift, C# and HTML5.

Mobile app development is rapidly growing. From retail, telecommunications and e-commerce to insurance, healthcare and government, organizations across industries must meet user expectations for real-time, convenient ways to conduct transactions and access information. Today, mobile devices—and the mobile applications that unlock their value—arethe most popular way for people and businesses to connect to the internet. To stay relevant, responsive and successful, organizations need to develop the mobile applications that their customers, partners and employees demand.

Yet mobile application development might seem daunting. Once you’ve selected the OS platform or platforms, you need to overcome the limitations of mobile devices and usher your app all the way past the potential hurdles of distribution. Fortunately, by following a fewbasic guidelines and best practices, you can streamline your application development journey.

1.1 Introduction to Project:

A cutting-edge event management app designed to streamline the planning and execution of all your events. Whether you're organizing a corporate conference, a wedding, or a casual meetup, this offers a comprehensive suite of features to ensure your event runs smoothly. From intuitive scheduling tools and real-time attendee tracking to seamless communication and vendor management, this simplifies every step of the process. The app's user-friendly interface and robust analytics capabilities allow event planners to make informed decisions and deliver memorable experiences. With this, managing events has never been easier or more efficient. The app provides intuitive scheduling tools that allow users to create detailed agendas and set reminders effortlessly. Real-time attendee tracking offers insights into event participation and engagement, while built-in messaging and notification systems facilitate seamless communication with attendees, vendors, and staff. decision-making.

1.2 Problem Statement:

The event management industry faces numerous challenges that can impede the successful planning and execution of events. Organizers often struggle with coordinating various aspects such as scheduling, attendee tracking, vendor management, and effective communication. Traditional methods, which may involve spreadsheets, emails, and phone calls, are inefficient and prone to errors. There is a need for a streamlined, integrated solution that simplifies these processes, enhances coordination, and provides actionable insights to improve the overall event experience. This is where an innovative event management app, like Event Ease, can address these pain points by offering a comprehensive suite of tools designed to make event planning and execution more efficient and effective.

1.3 Objectives:

The primary objective of our event management app is to revolutionize the way events are planned, organized, and executed. Our goal is to provide event planners and organizers with a powerful yet intuitive tool that simplifies every aspect of event management. This includes streamlining the scheduling process, enhancing attendee engagement through real-time tracking and communication features, facilitating seamless coordination with vendors and suppliers, and offering comprehensive analytics to measure event success and optimize future planning. By focusing on user experience and leveraging mobile technology, our app aims to make event management more efficient, cost-effective, and enjoyable, ultimately ensuring that every event is a resounding success. We envision our app as not just a tool, but a strategic partner in creatingmemorable and impactful events.

CHAPTER 2

SOFTWARE REQUIREMENT SPECIFICATION

A Software Requirements Specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. It outlines all functional and non- functional requirements, user interactions, constraints that ensure the software meets its goals.

2.1 Functional Requirements:

In software and systems engineering, functional requirement is a declaration of the intended function of a system and its components. Based on functional requirements, an engineer determines the behavior (output) that a device or software is expected to exhibit in the case of a certain input. A system design is an early form of a functional requirement.

1. User Registration and Authentication
2. Event Creation and Management
3. Scheduling and Agenda Management
4. Attendee Management
5. Communication Tools
6. Vendor and Supplier Management
7. Ticketing and Payment Processing
8. Venue Management
9. Attendee Engagement Features

2.2 Non-Functional Requirements:

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system. They are contrasted with functional requirements that define specific behaviour or functions.

1) Availability
2) Adaptability
3) Aesthetics
4) Compatibility
5) Consistency
6) Understandability

2.3 System Requirements:

Most of the computer games require high configurations of computer. But in the case of the proposed gaming system, the system requirements is not that much. The minimum systems requirements for the proposed project event management application" is mentioned following.

Hardware Requirements:

(a) Processor: 1.2 GHz

(b) RAM: 512MB

(c) Storage: 100 MB

Software Requirements:

(a) Operating System: Android 4.0

(b) Language: javascript ,c ,c++

CHAPTER 3

SYSTEM ANALYSIS & DESIGN

System analysis and design is a crucial phase in the software development lifecycle (SDLC) that focuses on understanding business needs, defining system requirements, and designing the architecture of the system to meet those requirements.

3.1 Existing System:

Event planning software plays a pivotal role by offering comprehensive functionalities that automate and simplify numerous aspects of event management. These include online registration and ticketing systems, attendee management tools, scheduling and agenda management features, and communication platforms for seamless interaction with attendees, sponsors, and vendors.

These platforms facilitate virtual event spaces, live streaming capabilities, interactive engagement tools like live chat and Q&A sessions, and robust analytics to measure attendee engagement and event success in the digital realm.

Mobile applications dedicated to event management provide attendees with personalized schedules, real-time updates, networking opportunities, and interactive maps, enhancing overall attendee experience and engagement. Venue management software assists in efficiently managing venue bookings, logistics, and catering arrangements, ensuring smooth operations on-site.

Despite the advancements, challenges remain, including the integration of diverse systems, ensuring data security and privacy, adapting to rapidly evolving technology, and meeting the specific needs and budgets of event organizers. However, the continuous innovation and adoption of these technologies underscore their importance in enhancing efficiency, attendee satisfaction, and the overall success of events in today's dynamic and competitive landscape.

3.2 PROPOSED SYSTEM:

The proposed event management system aims to integrate advanced technology and user-centric features to revolutionize how events are planned, organized, and executed. Key objectives include enhancing efficiency, improving attendee engagement, and providing comprehensive tools for seamless event management.

The system will feature a user-friendly interface with modules for event creation, scheduling, and agenda management, allowing organizers to effortlessly set up and manage event timelines, sessions, and activities. Robust attendee management capabilities will enable efficient registration processes, attendee tracking, and personalized communication through integrated messaging and notification systems.

To optimize vendor coordination, the system will include tools for managing contracts, payments, and logistics, ensuring smooth collaboration with event suppliers. Real-time analytics and reporting functionalities will provide valuable insights into attendee demographics, engagement metrics, and event performance, empowering organizers to make data-driven decisions and continuously improve event strategies.

Mobile compatibility will be a priority, ensuring that organizers and attendees can access critical event information and functionalities on the go. Integration with virtual event platforms will support hybrid event formats, facilitating live streaming, virtual networking, and interactive sessions to cater to diverse attendee preferences and global accessibility.

Security measures will be implemented to safeguard sensitive data and ensure compliance with data protection regulations. Customization options will allow organizers to tailor the system to their specific event requirements, branding, and organizational workflows, ensuring flexibility and scalability for events of varying sizes and complexities.

Overall, the proposed event management system aims to set new standards in efficiency, engagement, and effectiveness, empowering organizers to deliver seamless and memorable event experiences while maximizing operational efficiency and attendee satisfaction.

3.3 ARCHITECTURE &FLOW CHART

Architecture:

The architecture of a Event management app typically involves several key components and layers to ensure functionality, security, and usability.

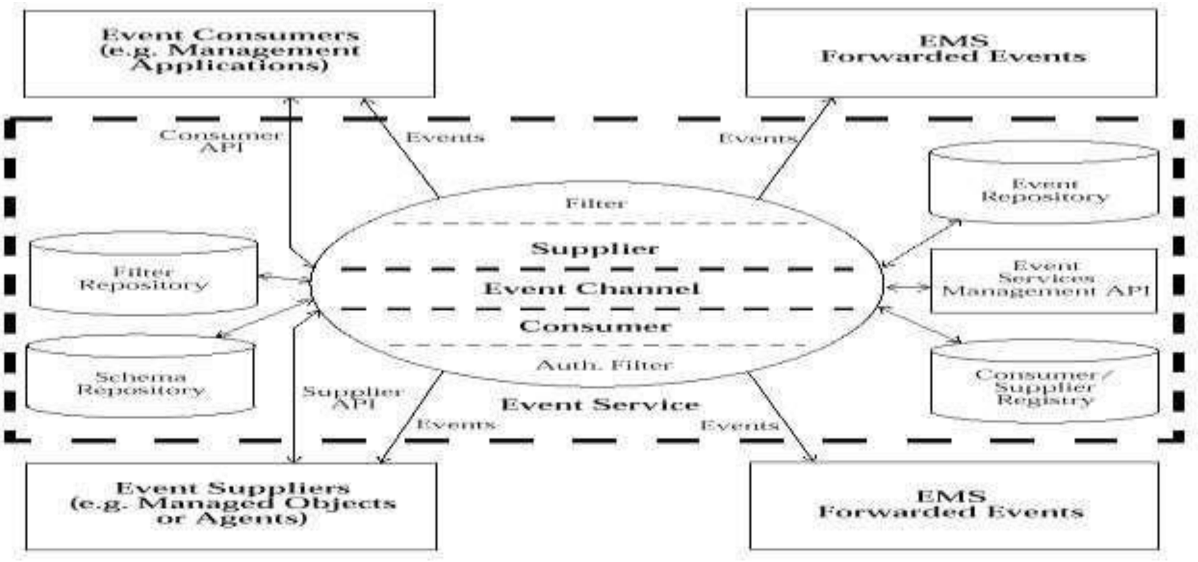


Fig.3.1 Architecture for Event management application

Designing the architecture for an event management app involves creating a robust system that seamlessly integrates frontend and backend components to deliver a smooth user experience. At the frontend, intuitive user interfaces are developed using modern frameworks like React or Angular, ensuring responsiveness across various devices. The backend typically comprises a scalable server application, such as Node.js or Django, coupled with a suitable database system like MongoDB to manage event details, user data, and transactions efficiently. Secure authentication mechanisms, such as JWT or OAuth, are implemented to safeguard user accounts and control access to features. Key functionalities include event creation with customizable details, seamless attendee registration and ticketing, and real-time notifications for updates and reminders. Integration with payment gateways like Stripe or PayPal facilitates secure transactions. To optimize performance, caching mechanisms and load balancers ensure swift data retrieval and distribution of server loads. Cloud deployment on platforms like AWS or Azure supports scalability and reliability, while adherence to data privacy regulations like GDPR ensures the protection of user information.

Sequence diagram:

Sequence diagrams are used to illustrate the dynamic behavior of a system by showing the interactions among various components, objects.

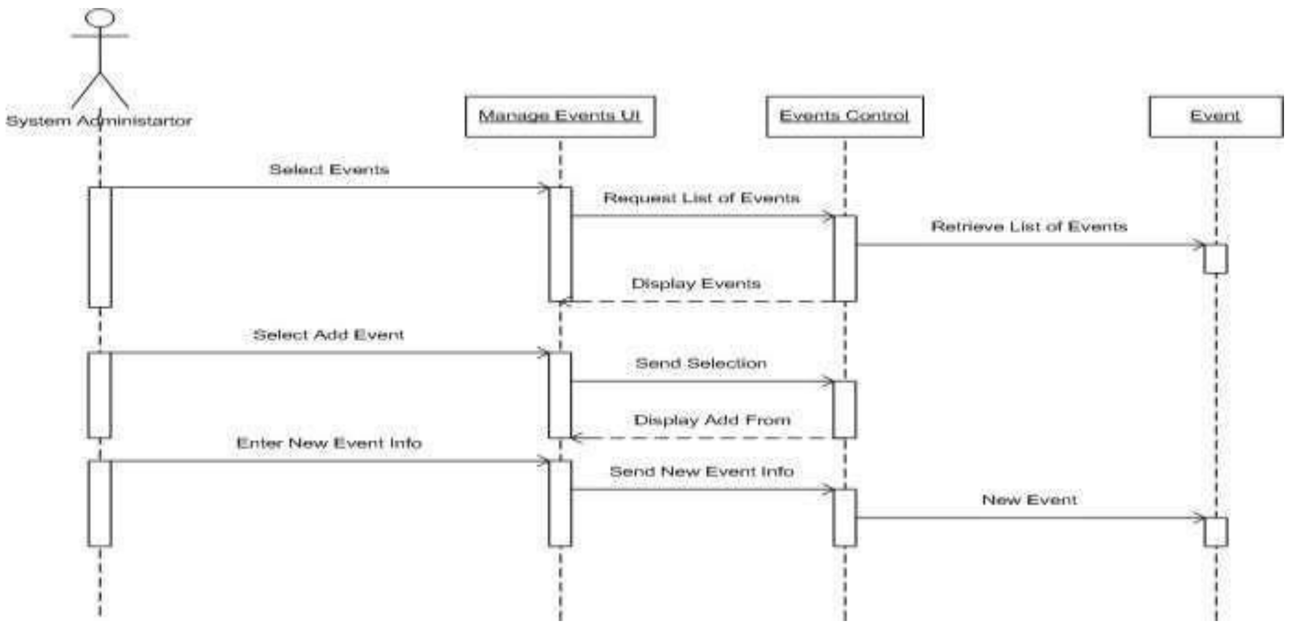


Fig 3.2 Sequence diagram

A sequence diagram for an event management app illustrates the interactions between various components and users during typical processes. Consider a scenario where a user creates and manages an event. The sequence begins with a user interacting with the frontend UI, initiating the process of creating a new event. The frontend sends a request to the backend server, passing details such as event title, description, location, date, and ticket option. The backend server receives the request and validates the data. It may communicate with external services, such as geolocation APIs for event location verification. Upon successful validation, the backend interacts with the database to store the event details securely. This involves writing entries for the event, ticket types, and associated pricing. Simultaneously, a notification service is triggered to inform the event organizer about successful event creation via email or push notification. When attendees register for the event through the app, the frontend sends registration requests to the backend. The server verifies availability and processes ticket purchases using integrated payment gateways. Upon successful registration, the frontend displays a confirmation message to the attendee, confirming their ticket purchase and providing event details. This sequence diagram outlines the flow of interactions between the frontend UI, backend server, database, notification services, payment gateways, and users.

CHAPTER 4

IMPLEMENTATION

Implementation is the execution or practice of a plan, a method or any design, idea, model, specification, standard or policy for doing something.

1.1 Pseudocode:

Pseudocode is a description of the steps in an algorithm using a mix of conventions of programming languages with informal.

CODE:

(a) MainActivity.java package

```
package com.sabikrahat.eventmanagementapp;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import  android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Toast;

public class LoginActivity extends AppCompatActivity {

    private EditText userIdEditText, userPwdEditText, userConfirmPwdEditText;
    private CheckBox rememberUserId, rememberLogin;

    boolean isCreating = false;

    SharedPreferences shp;
    SharedPreferences.Editor myEdit;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        shp = getSharedPreferences("MySharedPref", MODE_PRIVATE);

        myEdit = shp.edit();

        findViewById(R.id.exitBtn).setOnClickListener(view -> finish());

        userIdEditText = findViewById(R.id.userIdEditText);
        userPwdEditText = findViewById(R.id.pwdEditText);
        userConfirmPwdEditText = findViewById(R.id.rePwdEditText);
        rememberUserId = findViewById(R.id.rememberUserIdCheck);
        rememberLogin = findViewById(R.id.rememberLoginCheck);

        String userIdCheck = shp.getString("userId", "abc");
        if (userIdCheck.equals("abc")) {
            isCreating = true;
            userConfirmPwdEditText.setVisibility(View.VISIBLE);
        } else {
            if (shp.getBoolean("rememberLogin", false)) {
                isCreating = false;
                userConfirmPwdEditText.setVisibility(View.GONE);
                Toast.makeText(this, "Login Successfully", Toast.LENGTH_SHORT).show();
                startActivity(new Intent(getApplicationContext(), MainActivity.class));
                finish();
            }
        }

        findViewById(R.id.loginBtn).setOnClickListener(view -> userLogin());

        findViewById(R.id.noAccountId).setOnClickListener(view -> {
            if (isCreating) {
                isCreating = false;
                userConfirmPwdEditText.setVisibility(View.GONE);
            } else {
                isCreating = true;
                userConfirmPwdEditText.setVisibility(View.VISIBLE);
            }
        });
    }

    private void userLogin() {
        String userId = userIdEditText.getText().toString().trim();
        String userPwd = userPwdEditText.getText().toString().trim();
        String userRePwd = userConfirmPwdEditText.getText().toString().trim();
        boolean loginRemember = rememberLogin.isChecked();
        System.out.println("user Id: " + userId);
        System.out.println("user pwd: " + userPwd);
        System.out.println("user confirm Pwd: " + userRePwd);
        if (isCreating) {
            if (userId != "" && userPwd != "" && userRePwd != "") {
                if (userPwd.equals(userRePwd)) {
                    myEdit.putString("userId", userId);
                    myEdit.putString("userPwd", userPwd);
                    myEdit.putBoolean("rememberLogin", loginRemember);
                    myEdit.commit();

                    Toast.makeText(this, "Data Saved", Toast.LENGTH_SHORT).show();
                    startActivity(new Intent(getApplicationContext(), MainActivity.class));
                } else {
                    Toast.makeText(this, "Confirm Password doesn't match.",
Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText(this, "All fields required.", Toast.LENGTH_LONG).show();
            }
        } else {
            String id = shp.getString("userId", "");
            String pwd = shp.getString("userPwd", "");

            if (id == userId && pwd == userPwd) {
                Toast.makeText(this, "Login Successfully", Toast.LENGTH_SHORT).show();
                startActivity(new Intent(getApplicationContext(), MainActivity.class));
            } else {
                Toast.makeText(this, "user id & password invalid.", Toast.LENGTH_LONG).show();
            }
        }

    }
}

package com.sabikrahat.eventmanagementapp;

import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    // Reference objects for handling event lists
    private ListView lvEvents;
    private ArrayList<Event> events;
    private CustomEventAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        System.out.println("@MainActivity.onCreate");

        findViewById(R.id.createNewBtn).setOnClickListener(view -> {
            Intent intent = new Intent(MainActivity.this, EventInformation.class);
            startActivity(intent);
        });

        findViewById(R.id.exitBtn).setOnClickListener(view -> finish());

        findViewById(R.id.historyBtn).setOnClickListener(view -> startActivity(new
Intent(MainActivity.this, ShowWebViewActivity.class)));

        // initialize list-reference by ListView object defined in XML
        lvEvents = findViewById(R.id.lvEvents);
        // load events from database if there is any
        loadData();


    }

    private void loadData() {
        events = new ArrayList<>();
        KeyValueDB db = new KeyValueDB(this);
        Cursor rows = db.execute("SELECT * FROM key_value_pairs");
        if (rows.getCount() == 0) {
            return;
        }
        //events = new Event[rows.getCount()];
        while (rows.moveToNext()) {
            String key = rows.getString(0);
            String eventData = rows.getString(1);
            String [] fieldValues = eventData.split("-::-");

            String name = fieldValues[0];
            String place = fieldValues[1];
            String dateTime = fieldValues[2];
            String capacity = fieldValues[3];
            String budget = fieldValues[4];
            String email = fieldValues[5];
            String phone = fieldValues[6];
            String description = fieldValues[7];
            String eventType = fieldValues[8];

            Event e = new Event (key, name, place, dateTime, capacity, budget, email, phone,
description, eventType);
            events.add(e);
        }
        db.close();
        adapter = new CustomEventAdapter(this, events);
        lvEvents.setAdapter(adapter);

        // handle the click on an event-list item
        lvEvents.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, final View view, int position, long id)
{
                // String item = (String) parent.getItemAtPosition(position);
                System.out.println(position);

                Intent i = new Intent(MainActivity.this, EventInformation.class);
                i.putExtra("EventKey", events.get(position).key);
                startActivity(i);
            }
        });
        // handle the long-click on an event-list item
        lvEvents.setOnItemLongClickListener((parent, view, position, id) -> {
            //String message = "Do you want to delete event - "+events[position].name +" ?";
            String message = "Do you want to delete event - " + events.get(position).name + " ?";
            System.out.println(message);
            showDialog(message, "Delete Event", events.get(position).key);
            return true;
        });
    }
    private void showDialog(String message, String title, String key) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage(message);
        builder.setTitle(title);
        builder.setCancelable(false).setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int i) {
                Util.getInstance().deleteByKey(MainActivity.this, key);
                dialog.cancel();
                loadData();
                adapter.notifyDataSetChanged();
            }
        }).setNegativeButton("No", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int i) {
                dialog.cancel();
            }
        });

        AlertDialog alert = builder.create();
        alert.show();
    }

    @Override
    public void onStart() {
        super.onStart();
        System.out.println("@MainActivity.onStart");
    }

    @Override
    public void onResume() {
        super.onResume();
        System.out.println("@MainActivity.onResume");
    }

    @Override
    public void onPause() {
        super.onPause();
        System.out.println("@MainActivity.onPause");
    }

      }
   @Override
      public void onRestart() {
          super.onRestart();
          System.out.println("@MainActivity.onRestart");
          // re-load events from database after coming back from the next page
          loadData();
      }

      @Override
      public void onStop() {
          super.onStop();
          System.out.println("@MainActivity.onStop");
          // clear the event data from memory as the page is completely hidden by now
          events.clear();
      }

      @Override
      public void onDestroy() {
          super.onDestroy();
          System.out.println("@MainActivity.onDestroy");
      }
   }
```

CHAPTER 5
TESTING

Testing is used to check the validation of the application.

- Check is the Functionality Testing
- Check is the Usability Testing
- Check is the Security Testing
- Check is the Compatibility Testing
- Check is the Integration Testing
- Check is the Regression Testing
- Check is the Accessibility Testing
- Check is the Localization Testing
- Check is the User Acceptance Testing (UAT)

Test Scenarios for Event Management System:

Testing scenarios for an event management app cover various aspects to ensure its functionality, usability, and reliability across different user interactions. Here are key testing scenarios:

1. Event Creation and Management:
   - Verify that organizers can successfully create events with all necessary details such as title, description, date, time, location, and ticket types.
   - Test edge cases like creating events with long titles or descriptions to ensure input validation works correctly.
   - Ensure organizers can edit event details after creation and that changes are reflected accurately.

2. Registration and Ticketing:
   - Test the attendee registration process to ensure users can select tickets, provide required information.
   - Validate that attendees receive confirmation emails or notifications after registering for an event.
   - Verify that ticket availability updates in real-time as tickets are purchased.

3. Payment Integration:

- Verify different scenarios such as successful payments, declined payments, and refunds if applicable.
- Ensure payment details are handled securely and sensitive information is encrypted.

4. Notification System:
- Test the functionality of notification services (email, push notifications) to ensure organizers.
- Verify that notifications are triggered correctly based on user actions (e.g., event creation,)

5. User Authentication and Authorization:
- Validate that user authentication (login, logout) works securely using methods like JWT or OAuth.
- Test role-based access control to ensure organizers have appropriate permissions to create.

6. Performance and Load Testing:
- Conduct load tests to simulate concurrent user activity during peak times (e.g., ticket sales) .
- Measure response times for critical actions such as event creation, registration, and ticket purchase

7. Security Testing:
- Perform security tests to identify vulnerabilities such as SQL injection, cross-site scripting (XSS).
- Validate data encryption mechanisms for storing and transmitting user information and payment.

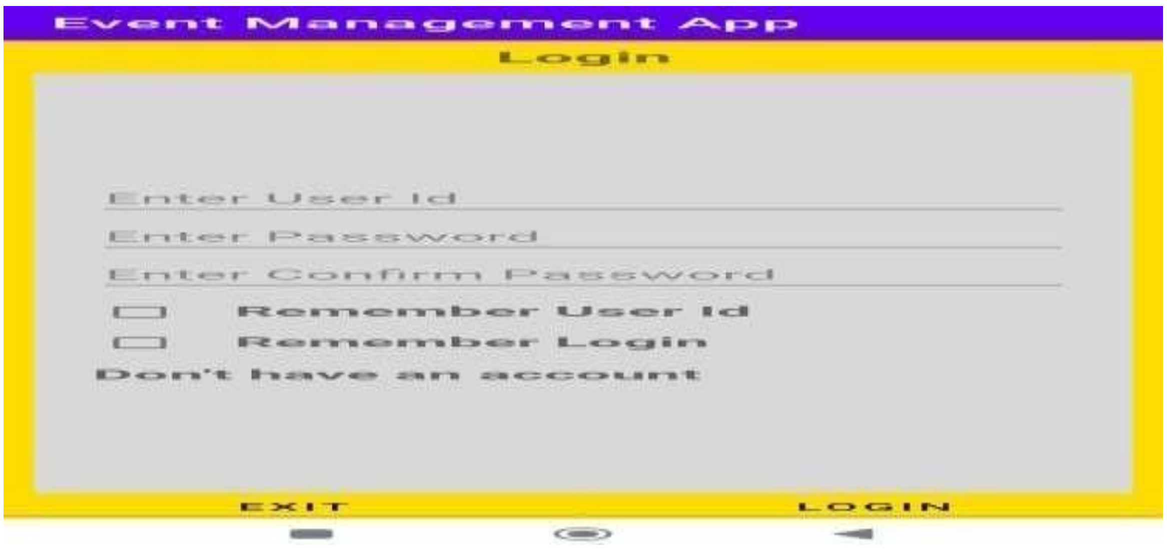
8. Usability and Accessibility Testing:
- Evaluate the app's user interface (UI) for intuitiveness, responsiveness, and ease of navigation .
- Conduct accessibility tests to ensure compliance with accessibility standards (WCAG) for users .

CHAPTER 6
RESULTS AND SNAPSHOTS

Outcomes or outputs derived from a process, action, or series of actions. Here, there is a snapshot of application.

5.1 Front page of project:

This page is the front page of the application.



Event Management App

Login

Enter User Id

Enter Password

Enter Confirm Password

☐ Remember User Id

☐ Remember Login

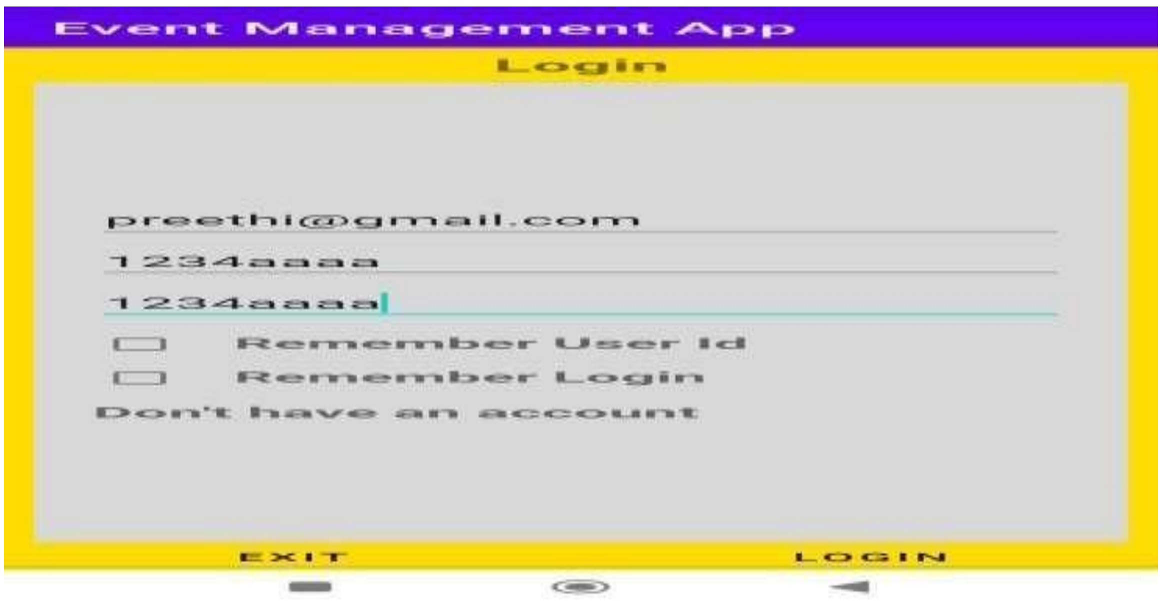
Don't have an account

EXIT LOGIN

Fig 6.1: Front page of project

5.2 Login page:

This page is used to login to the app.



Event Management App

Login

preethi@gmail.com

1234aaaa

1234aaaa

☐ Remember User Id

☐ Remember Login

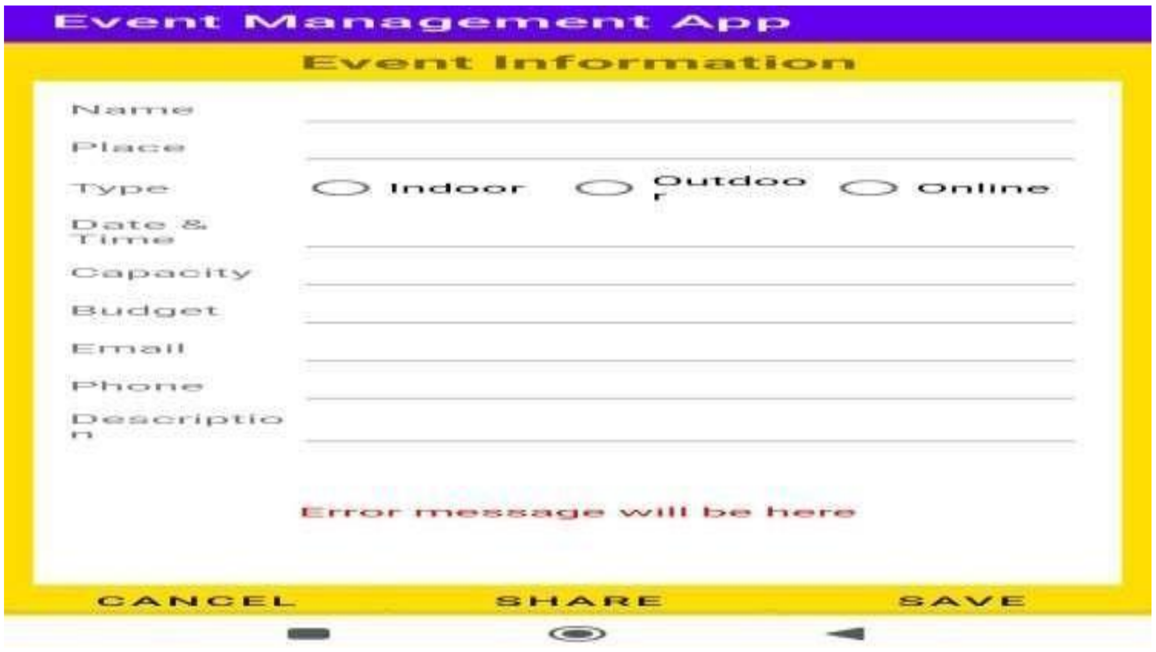
Don't have an account

EXIT LOGIN

Fig 6.2: Login page

6.3 Event information page:

This page is used to enter the information about the event.



Event Management App

Event Information

Name

Place

Type ◯ Indoor ◯ Outdoor ◯ Online

Date & Time

Capacity

Budget

Email

Phone

Description

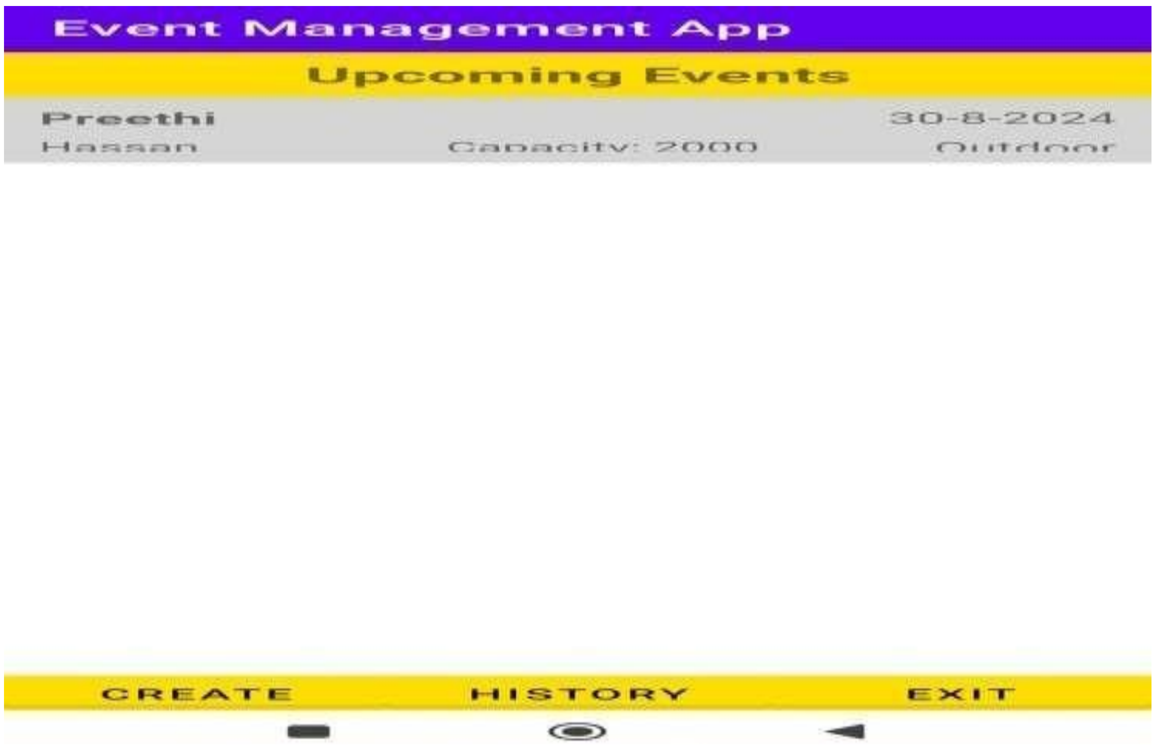
Error message will be here

CANCEL SHARE SAVE

Fig 6.3: Event information

6.4 Upcoming events:

This page is used to show the list of upcoming events.



Event Management App

Upcoming Events

Preethi		30-8-2024
Hassan	Capacity: 2000	Outdoor

CREATE HISTORY EXIT

Fig 6.4: Upcoming events

CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION

Concluding the event management app involves a comprehensive review process aimed at assessing its effectiveness and user satisfaction. This includes evaluating how well the app streamlined event planning, registration, communication, and attendee management. Gathering feedback from organizers and participants is crucial to identifying strengths and areas needing improvement. Analyzing usage data provides insights into user engagement, feature usage, and technical performance. Updates and enhancements based on this feedback ensure the app evolves to meet user needs effectively. Documenting lessons learned and successes achieved during deployment prepares for future iterations or similar projects. Ultimately, a thorough conclusion of the app involves celebrating achievements and planning for ongoing support to maintain its impact and relevance in the event management landscape.

FUTURE ENHANCEMENT

In envisioning future enhancements for an event management app, several transformative features can elevate the user experience and operational efficiency. Integration with Virtual Reality (VR) and Augmented Reality (AR) could revolutionize venue selection and event planning, allowing users to visualize and customize event setups in real-time immersive environments. Enhanced analytics capabilities could provide deeper insights into attendee demographics, engagement metrics, and feedback, facilitating data-driven decision-making and improving event outcomes. Furthermore, incorporating sustainability features, such as carbon footprint tracking and eco-friendly vendor options, would cater to growing environmental concerns in event planning. These enhancements aim not only to streamline event logistics but also to enrich attendee experiences through personalization, interactive agendas, and innovative engagement strategies like gamification and live streaming support for hybrid events, ensuringthe app remains at the forefront of event management technology.

REFERENCES

- Elberse, A., & Gupta, S. (2010). Event management system*: Time to Charge Users?*. Cambridge, MA: Harvard Business Review.
- Kim, J. (2012). The institutionalization of event management system: From user-generated content to professionally generated content. *Media, Culture & Society*, *34*(1), 53–67. Retrieved March 9, 2015, from http://mcs.sagepub.com/content/34/1/53.full
- Strangelove, M. (2010)*:* Extraordinary Videos by Ordinary *People*. Toronto: University of Toronto Press.
- Top event management by Subscribed. (n.d.). Retrieved March 4, 2015, from https://socialblade.com/youtube/top/100/mostsubscribed