Roll No: | | | | | | | | | | | |

PRESIDENCY UNIVERSITY

GAIN MORE KNOWLEDGE
REACH GREATER HEIGHTS

Presidency University Act, 2013 of the Karnataka Act No. 41 of 2013 | Established under Section 2(f) of UGC Act, 1956
Approved by AICTE, New Delhi | Approved By BCI
Bengaluru

Mid Term Examinations - October 2026

Date: 05/10/2026 **Time:** 11:45 AM - 01:15 PM

Course Code: CSD3421 **Course Name:** Mobile Application for Data Science

Semester: Seventh Semester **Max. Marks:** 50 **Weightage:** 50%

CO - Levels	CO1	CO2	CO3
Marks	24	44	22

ANSWER KEY

PART A: Answer Following Questions. 10 M

Q.N.	Questions	M	CO	BT
1	Cross-platform mobile application development is the process of developing a single mobile application that can run on multiple operating systems, such as Android and iOS, using a shared codebase. Example: Flutter allows developers to build Android and iOS applications using the Dart programming language, reducing the need to develop separate applications for each platform.	2	CO1	BT1
2	The Flutter SDK provides the tools, libraries, and components required to develop, test, and run Flutter applications.	2	CO1	BT1
3	Form validation in Flutter is the process of checking user inputs in a form to ensure they are correct, complete, and meet specified requirements before the data is submitted. For example, it can check whether an email address is valid, a password meets the required length, or a name field is not empty. Flutter uses the Form and TextFormField widgets, along with a validator function, to perform form validation.	2	CO2	BT1
4	Responsive UI development is the process of designing an interface that adapts to different screen sizes and orientations.	2	CO2	BT2
5	The appropriate status code is: 404 – Not Found It indicates that the requested resource could not be found on the server. Example: HTTP/1.1 404 Not Found The application can display an appropriate message such as: Sensor record not found.	2	CO3	BT3

PART B : Answer Any 1 Following Questions. 10 M

Q.N.	Questions	M	CO	BT

6	Flutter is an open-source UI framework developed by Google for building cross-platform applications from a single codebase. Flutter Architecture Flutter architecture consists of three major layers: Framework Layer Provides widgets, animation, rendering, and other UI-related APIs. Written primarily in Dart. Helps developers design application interfaces. Flutter Engine Responsible for rendering graphics and text. Provides low-level functionality such as graphics, text layout, and runtime support. Embedder Connects the Flutter engine to the target operating system. Handles platform-specific integration, input, and application lifecycle. Role of Dart Dart is the programming language used to develop Flutter applications. It is used to write UI components and application logic. It supports object-oriented programming. It enables features such as hot reload during development. Role of Flutter SDK Provides Flutter libraries and development tools. Includes tools to create, build, test, and run applications. Supports deployment to platforms such as Android, iOS, web, and desktop. Conclusion: Flutter combines the Dart-based framework, rendering engine, and platform embedder to support cross-platform application development.	10	CO1	BT2

ANSWER KEY

7	Object-Oriented Programming in Dart – 2 Marks
	
	Object-Oriented Programming (OOP) is a programming paradigm in which a program is organized around objects that contain data and methods that operate on that data.
	
	Dart supports important OOP concepts such as:
	
	Classes
	Objects
	Constructors
	Encapsulation
	Inheritance
	Polymorphism
	Abstraction
	
	OOP helps in developing applications that are modular, reusable, maintainable, and easier to extend.
	
	Marking:
	
	Definition of OOP – 1 mark
	Mentioning relevant OOP features/advantages – 1 mark
	Classes in Dart – 2 Marks
	
	A class is a blueprint or template for creating objects. It defines the properties (data members) and methods (behaviour) that objects of that class will have.
	
	Example:
	class Student {
	String name;
	int age;
	
	void displayDetails() {
	print("Name: $name");
	print("Age: $age");
	}
	}
	Correct definition of class – 1 mark
	Appropriate Dart example and identification of members – 1 mark
	Constructors in Dart – 3 Marks
	
	A constructor is a special method used to initialize an object when it is created.
	
	A constructor has the same name as the class.
	
	Default/Generative Constructor
	class Student {
	String name;
	int age;
	
	Student(this.name, this.age);
	
	void displayDetails() {
	print("Name: $name");
	print("Age: $age");
	}
	}
	Definition/purpose of constructor – 1 mark
	Correct constructor syntax – 1 mark
	Correct use of constructor while creating objects – 1 mark
	A complete example combining class, object, properties, method, and constructor is:
	class Student {
	String name;
	int age;
	
	Student(this.name, this.age);
	
	void displayDetails() {
	print("Name: $name");
	print("Age: $age");
	}
	}
	
	void main() {
	Student student1 = Student("Anu", 21);
	Student student2 = Student("Rahul", 22);
	
	student1.displayDetails();
	student2.displayDetails();
	}
		10	CO1	BT2

ANSWER KEY

PART C: Answer Any 1 Following Questions. 10 M				
Q.N.	**Questions**	**M**	**CO**	**BT**
8	Material Design is a design system that provides guidelines for creating consistent, attractive, and user-friendly interfaces. Key principles Consistency: Similar components follow a common design pattern. Visual hierarchy: Important content is emphasized through size, color, and placement. Meaningful feedback: UI elements respond to user interactions. Navigation: Users can move between screens in a predictable way. Responsive layouts: Interfaces adapt to different screen sizes. Material Design in Flutter Flutter provides Material widgets such as: Scaffold – Provides a basic page structure. AppBar – Displays the top application bar. ElevatedButton – Creates a raised button. TextField – Accepts user input. Card – Displays content in a card-like layout. Example: Scaffold(appBar: AppBar(title: const Text("My Application"),), body: Center(child: ElevatedButton(onPressed: () {}, child: const Text("Click Me"),),),); Conclusion: Material Design widgets simplify UI development and help create consistent Flutter applications. OR Q8 (b). Explain responsive UI development and theme management in Flutter. (10 Marks – BL2) Answer: Responsive UI Development Responsive UI development allows an application to adapt to different screen sizes, orientations, and device layouts. Flutter supports responsive design using: MediaQuery – Provides information about screen dimensions. LayoutBuilder – Builds layouts based on available constraints. Expanded – Allows a child to occupy available space. Flexible – Allows flexible sizing within a layout. Theme Management Theme management ensures a consistent appearance throughout the application. Flutter uses ThemeData to define common styling properties such as: Primary and secondary colors Text styles Button styles Background colors MaterialApp(theme: ThemeData(colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue,),), home: const MyHomePage(),);	10	CO2	BT2

ANSWER KEY

9	Flutter uses a widget-based approach for designing user interfaces. Row, Column, Stack, and Container are commonly used layout widgets that help arrange, position, and style UI elements. 1. Row Widget – 2 Marks A Row arranges its child widgets horizontally, along the main axis. Syntax: Row(children: [Icon(Icons.home), Text("Home"), Icon(Icons.arrow_forward),],) The widgets are placed from left to right. Important properties include: children – widgets displayed inside the Row mainAxisAlignment – controls horizontal alignment crossAxisAlignment – controls vertical alignment mainAxisSize – controls the amount of horizontal space occupied Example: Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly, children: [Text("Home"), Text("Profile"), Text("Settings"),],) Definition and horizontal arrangement – 1 mark Correct example/properties – 1 mark Column Widget – 2 Marks A Column arranges its child widgets vertically, along the main axis. Example: Column(children: [Text("Name"), Text("Email"), ElevatedButton(onPressed: () {}, child: Text("Submit"),),],) The widgets are arranged from top to bottom. Important properties include: children mainAxisAlignment crossAxisAlignment mainAxisSize Example: Column(mainAxisAlignment: MainAxisAlignment.center, crossAxisAlignment: CrossAxisAlignment.center, children: [Text("Welcome"), Text("Login to continue"),],) Definition and vertical arrangement – 1 mark Correct example/properties – 1 mark Stack Widget – 2 Marks A Stack allows widgets to be placed on top of one another. It is useful when UI elements need to overlap. For example, an image can be displayed with text or an icon over it. Example: Stack(children: [Image.network("https://example.com/image.jpg",), Positioned(bottom: 10, left: 10, child: Text("Profile Photo", style: TextStyle(color: Colors.white),),),],) Definition and overlapping concept – 1 mark Correct example using Stack/Positioned – 1 mark Container Widget – 2 Marks A Container is commonly used to provide padding, margin, size, alignment, decoration, and background styling to a widget. Example: Container(width: 200, height: 100, padding: EdgeInsets.all(16), alignment: Alignment.center, decoration: BoxDecoration(color: Colors.blue, borderRadius: BorderRadius.circular(10),), child: Text("Hello Flutter",),) A Container can therefore control the appearance and layout of its child. Common properties include: width height padding margin alignment decoration color child Definition and purpose – 1 mark Correct example/properties – 1 mark Combined Use of Row, Column, Stack and Container A Flutter interface generally combines multiple layout widgets. For example: Container(padding: EdgeInsets.all(16), child: Column(children: [Stack(children: [Container(width: 300, height: 150, color: Colors.blue,), Positioned(bottom: 10, left: 10, child: Text("Data Science"),),],), Row(mainAxisAlignment: MainAxisAlignment.spaceEvenly, children: [Text("Analytics"), Text("Reports"), Text("Profile"),],),],),)	10	CO2	BT2

ANSWER KEY

PART D: Answer Any 1 Following Questions. 10 M				
Q.N.	**Questions**	**M**	**CO**	**BT**

ANSWER KEY

10	import 'package:flutter/material.dart';
	
	void main() {
	runApp(const MyApp());
	}
	
	class MyApp extends StatelessWidget {
	const MyApp({super.key});
	
	@override
	Widget build(BuildContext context) {
	return MaterialApp(
	debugShowCheckedModeBanner: false,
	home: const GreetingPage(),
	);
	}
	}
	
	class GreetingPage extends StatefulWidget {
	const GreetingPage({super.key});
	
	@override
	State createState() => _GreetingPageState();
	}
	
	class _GreetingPageState extends State {
	final TextEditingController nameController = TextEditingController();
	
	String greeting = "";
	
	void displayGreeting() {
	setState(() {
	greeting = "Hello, ${nameController.text}!";
	});
	}
	
	@override
	void dispose() {
	nameController.dispose();
	super.dispose();
	}
	
	@override
	Widget build(BuildContext context) {
	return Scaffold(
	appBar: AppBar(
	title: const Text("Greeting App"),
	),
	body: Padding(
	padding: const EdgeInsets.all(20.0),
	child: Column(
	mainAxisAlignment: MainAxisAlignment.center,
	children: [
	TextField(
	controller: nameController,
	decoration: const InputDecoration(
	labelText: "Enter your name",
	border: OutlineInputBorder(),
	),
	),
	
	const SizedBox(height: 20),
	
	ElevatedButton(
	onPressed: displayGreeting,
	child: const Text("Greet"),
	),
	
	const SizedBox(height: 20),
	
	Text(
	greeting,
	style: const TextStyle(
	fontSize: 24,
	fontWeight: FontWeight.bold,
	),
	),
	],
	),
	),
	);
	}
	}
		10	CO2	BT3

ANSWER KEY

11	import 'package:flutter/material.dart';
	
	void main() {
	runApp(const MyApp());
	}
	
	class MyApp extends StatelessWidget {
	const MyApp({super.key});
	
	@override
	Widget build(BuildContext context) {
	return MaterialApp(
	home: Scaffold(
	appBar: AppBar(
	title: const Text('Student Details'),
	),
	body: const Center(
	child: Column(
	mainAxisAlignment: MainAxisAlignment.center,
	children: [
	Text('Name: Anu', style: TextStyle(fontSize: 20)),
	Text('Register No: 101', style: TextStyle(fontSize: 20)),
	Text('Department: CSE', style: TextStyle(fontSize: 20)),
	],
	),
	),
	),
	);
	}
	}
		10	CO2	BT3

PART E: Answer Any 1 Following Questions. 10 M

Q.N.	Questions	M	CO	BT
12	Appropriate resource-oriented URIs | Operation | URI | | ---------- | ---------- | | Retrieve all students | `/api/v1/students` | | Retrieve student 101 | `/api/v1/students/101` | | Add a student | `/api/v1/students` | | Completely update student 101 | `/api/v1/students/101` | | Partially update CGPA | `/api/v1/students/101` | | Delete student 101 | `/api/v1/students/101` | HTTP methods – 3 Marks | Operation | HTTP Method | | ---------- | ---------- | | Retrieve all students | GET | | Retrieve particular student | GET | | Add student | POST | | Completely update student | PUT | | Partially update CGPA | PATCH | | Delete student | DELETE | Query parameter – 1 Mark /api/v1/students?department=CSE Nouns instead of verbs – 1 Mark REST APIs should use resource-oriented nouns. Preferred: /students Instead of: /getStudents Statelessness – 2 Marks In a stateless REST API, every HTTP request contains the information required by the server to process that request. The server does not depend on stored client-session state between requests. Request 1 → contains authentication information Request 2 → contains authentication information Request 3 → contains authentication information	10	CO3	BT3

ANSWER KEY

	HTTP method and endpoint – 1 Mark
	POST /api/v1/sensordata
	Full endpoint:
	https://api.example.com/api/v1/sensordata
	HTTP headers – 2 Marks
	Answer:
	
	Two appropriate headers are:
	Content-Type: application/json
	Authorization: Bearer
	Content-Type with explanation – 1 mark
	Authorization with explanation – 1 mark
	SON request body – 2 Marks
	Answer:
	{
	"deviceId": "AND-2201",
	"temperature": 28.5
	}
	
	Correct deviceId – 1 mark
	Correct temperature and JSON syntax – 1 mark
	Successful HTTP response – 2 Marks
	Answer:
	
	A possible successful response is:
	
	HTTP/1.1 201 Created
	Content-Type: application/json
	
	Response body:
	{
	"id": 101,
	"status": "saved"
	}			

13	The 201 Created status indicates that the resource/data has been successfully created. Correct success status – 1 mark Appropriate JSON response body – 1 mark Handling 401, 404 and 503 – 3 Marks 401 – Unauthorized Indicates an authentication problem. The application can: Request the user to authenticate again. Refresh/reacquire a token where appropriate. Avoid repeatedly sending an invalid credential. 1 mark 404 – Not Found Indicates that the requested resource/endpoint could not be found. The application can display an appropriate error message or handle the missing resource. 1 mark 503 – Service Unavailable Indicates that the server/service is temporarily unavailable. The application can: Retry after an appropriate delay. Use exponential backoff. Use cached/local data if available. 1 mark	10	CO3	BT3

ANSWER KEY