Sarah Riley (Individual Project)

Student Information

- Rileys14@mymail.nku.edu

Planning the Individual Project

Summary of the Project

A mobile application to help students manage and track their time by recording tasks, specifying dates, times, and tags, and querying these records. This requires a simple user interface for time tracking with fields for date, start and end time, task description, and tag. We also want students to query their past activities based on the date, task, or tag.

Goals

Develop a mobile time-tracking application using Flutter for the front end and SQLite for backend services, allowing users to record tasks with flexible date and time inputs, tag them, and query the recorded tasks using customizable filters.

Why?

Students want to improve their time management by logging and tracking how they spend their time on specific tasks and activities. They seek an app with flexibility in recording times and tasks, along with the ability to query the data for reviewing past activities, which will help them analyze and improve their productivity.

Features

1. **Task Recording Screen**

- A simple form where users can input:
  - **Date**: Record the date of the task.
  - **From**: Start time of the task.
  - **To**: End time of the task.
  - **Task**: Description of the activity (e.g., "Studied Java").
  - **Tag**: A category for the task (e.g., "STUDY").

2. **Task Query Screen**

- A search feature where users can query past tasks based on:

- **Date**: Find tasks from a specific date.
- **Task**: Search by specific activity (e.g., "Java").
- **Tag**: Filter tasks by a category (e.g., "STUDY").

3. **SQLite Integration**

- Store and retrieve tasks from SQLite to manage user data securely.

4. **Basic User Authentication**

- Allow users to log in and manage their tasks using SQLite Authentication

Project Test Plan

1. Task Recording

Test Case 2.1: Record a Task with Valid Inputs

- **Objective**: Verify that tasks can be recorded with correct inputs.
- **Steps**:
  1. Log in and navigate to the task recording screen.
  2. Enter valid inputs for date, start time, end time, task, and tag.
  3. Submit the form.
- **Expected Result**: The task should be saved in the database, and a confirmation message should appear.

Test Case 2.2: Record a Task with Invalid Time Format

- **Objective**: Ensure the app handles invalid time formats correctly.
- **Steps**:
  1. Enter an invalid time format (e.g., "25:00").
  2. Submit the form.
- **Expected Result**: The form should not be submitted, and an error message should indicate the invalid time format.

Test Case 2.3: Record a Task with Missing Inputs

- **Objective**: Ensure the app prevents task recording if required fields are missing.
- **Steps**:
  1. Leave one or more fields blank (e.g., Task or Date).
  2. Submit the form.
- **Expected Result**: The form should not be submitted, and an error message should indicate missing fields.

2. Task Querying

Test Case 3.1: Query Tasks by Date

- **Objective**: Ensure tasks can be retrieved based on a specific date.
- **Steps**:
  1. Navigate to the task query screen.
  2. Enter a valid date and submit the form.
- **Expected Result**: The list of tasks for the given date should appear.

Test Case 3.2: Query Tasks by Task Name

- **Objective**: Ensure tasks can be retrieved based on a specific task name.
- **Steps**:
  1. Navigate to the task query screen.
  2. Enter a task keyword (e.g., "Java") and submit the form.
- **Expected Result**: All tasks containing that keyword should appear.

Test Case 3.3: Query Tasks by Tag

- **Objective**: Ensure tasks can be retrieved based on a specific tag.
- **Steps**:
  1. Navigate to the task query screen.
  2. Enter a tag (e.g., "STUDY") and submit the form.
- **Expected Result**: All tasks with that tag should appear.

3. SQLite Integration

Test Case 4.1: Data Syncing with SQLite

- **Objective**: Verify that tasks are properly stored and retrieved from SQLite.
- **Steps**:
  1. Check SQLite to confirm the task is stored.
  2. Query the task from the app.
- **Expected Result**: The task should be stored in SQLite and retrievable from the app.

Test Case 4.2: Data Sync after Reconnection

- **Objective**: Ensure the app can sync data after going offline and reconnecting.
- **Steps**:
  1. Record a task offline.
  2. Reconnect and check if the task syncs with SQLite.
- **Expected Result**: The task should sync with SQLite once the connection is restored.

4. General Usability

Test Case 5.1: Navigation Between Screens

- **Objective**: Ensure users can navigate between screens (e.g., task recording, task querying, and login screens).
- **Steps**:
  1. Log in and navigate through the app's screens.
- **Expected Result**: Users should be able to navigate smoothly between different sections of the app without errors.

Test Case 5.2: Error Messages Display Correctly

- **Objective**: Verify that appropriate error messages are displayed when invalid data is entered.
- **Steps**:
  1. Enter invalid data (e.g., incorrect date or time).
- **Expected Result**: An appropriate error message should be displayed without crashing the app.

Milestones and Deadline

1. Project Planning and Setup

- **Start Date**: 10/11/2024
- **End Date**: 10/13/2024
- **Milestone**: Finalize project requirements and set up the development environment.
- **Deliverables**: Detailed project scope, requirements, and tools (Flutter, SQLite) set up for development.

2. Basic UI/UX Design

- **Start Date**: 10/14/2024
- **End Date**: 10/17/2024
- **Milestone**: Create initial wireframes and mockups for the task recording, querying screens, and login page.
- **Deliverables**: Low-fidelity designs.
- **Note**: **Prototype Phase**: These wireframes will be part of an early prototype for internal review.

3. SQLite Integration

- **Start Date**: 10/18/2024
- **End Date**: 10/22/2024
- **Milestone**: Set up SQLite for authentication and Firestore for task data storage.
- **Deliverables**: SQLite services integrated and working with the app backend.

4. Prototype Version 1 Development: Basic Task Recording Functionality

- **Start Date**: 10/23/2024

- **End Date**: 10/29/2024
- **Milestone**: Implement the core task recording feature (date, time, task, and tag input) and store it in SQLite.
- **Deliverables**: Working prototype where users can record tasks and view data in SQLite.
- **Note**: **Prototype Version 1** will be completed after this stage, focusing on essential functionality.

5. User Authentication (SQLite Authentication)

- **Start Date**: 10/30/2024
- **End Date**: 11/03/2024
- **Milestone**: Implement user login, registration, and logout functionality.
- **Deliverables**: Secure user authentication through SQLite.
- **Tests**: Test user login, registration, and invalid credentials.

6. Task Query Functionality (Version 1)

- **Start Date**: 11/04/2024
- **End Date**: 11/08/2024
- **Milestone**: Implement basic query functionality where users can filter tasks by date, task, and tag.
- **Deliverables**: Users can search tasks based on criteria and retrieve records.
- **Tests**: Query by date, task, and tag filters to ensure correct results.

7. Prototype Version 2 Development: Data Validation and Error Handling

- **Start Date**: 11/09/2024
- **End Date**: 11/12/2024
- **Milestone**: Add input validation (e.g., date/time formats) and basic error handling for invalid inputs or missing fields.
- **Deliverables**: Robust data validation for task recording and querying.
- **Tests**: Test scenarios with invalid data (e.g., wrong time format or missing fields).

8. UI/UX Finalization (Version 2)

- **Start Date**: 11/13/2024
- **End Date**: 11/16/2024
- **Milestone**: Refine and finalize the UI/UX based on feedback from internal reviews of the prototype.
- **Deliverables**: A polished interface ready for end-user interaction.
- **Tests**: Perform usability testing with a small user group to identify pain points and improve UI/UX.

9. SQLite Sync and Offline Mode (Version 2)

- **Start Date**: 11/17/2024
- **End Date**: 11/20/2024

- **Milestone**: Implement offline mode, ensuring tasks can be recorded without a connection and synced when online.
- **Deliverables**: Full offline functionality and reliable data syncing with SQLite.
- **Tests**: Test offline data entry and check sync functionality when the connection is restored.

10. Final Version Development (Version 3): Polishing and Finalizing Core Features

- **Start Date**: 11/21/2024
- **End Date**: 11/26/2024
- **Milestone**: Complete the final version of the application, focusing on polishing the UI and ensuring all core features (task recording, querying, syncing) work seamlessly.
- **Deliverables**: Final version of the application ready for internal review.
- **Tests**: End-to-end testing, including task recording, querying, data validation, and sync across devices.

11. Final Testing, Bug Fixing, and Final Version Release

- **Start Date**: 11/27/2024
- **End Date**: 11/30/2024
- **Milestone**: Conduct final testing and fix any remaining bugs. Perform a final review to ensure everything works as expected.
- **Deliverables**: Final version of the app, thoroughly tested and ready for deployment.

12. Deployment

- **Start Date**: 12/01/2024
- **Milestone**: Application is fully developed and deployed.
- **Deliverables**: Project completed and application live.

Risk Analysis**1. Technical Risks**

1.1 SQLite Integration Issues

- **Risk**: Difficulty integrating SQLite services (Firestore, Authentication, etc.) with Flutter, which could lead to delays or malfunctioning features (e.g., task storage or user authentication).
- **Impact**: High
- **Likelihood**: Medium
- **Mitigation**: Follow SQLite and Flutter documentation closely, perform early testing on SQLite integration, and implement mock data for initial testing to isolate SQLite-related issues.

1.2 Data Syncing Issues

- **Risk**: The app may not sync offline data correctly once an internet connection is restored, leading to data loss or duplication.
- **Impact**: High

- **Likelihood**: Medium
- **Mitigation**: Thorough testing of offline and online sync features, using SQLite's offline persistence options, and implementing checks for potential data conflicts.

1.3 Platform-Specific Bugs

- **Risk**: Since the app is meant for both iOS and Android, platform-specific bugs may arise, causing inconsistent behavior across devices.
- **Impact**: Medium
- **Likelihood**: High
- **Mitigation**: Conduct separate testing for both platforms (iOS and Android), and use Flutter's platform-specific functionalities carefully to minimize issues.

1.4 Performance and Scalability

- **Risk**: The app may experience performance issues if a large amount of data is stored or queried, especially with complex query operations.
- **Impact**: Medium
- **Likelihood**: Medium
- **Mitigation**: Optimize SQLite queries and data structure. Test the application with a large data set early in development to identify performance bottlenecks.

2. Project Management Risks

2.1 Scope Creep

- **Risk**: Additional features or complexities might be added during development, causing delays or exceeding the budget.
- **Impact**: High
- **Likelihood**: Medium
- **Mitigation**: Define the project's scope clearly from the start and stick to the core features. Use agile project management to regularly assess and prioritize features.

2.2 Time Overruns

- **Risk**: The development timeline might extend due to unforeseen challenges, such as technical issues, team availability, or other delays.
- **Impact**: High
- **Likelihood**: Medium
- **Mitigation**: Set realistic timelines with built-in buffers. Conduct frequent progress reviews and adjust the timeline if needed.

2.3 Limited Resources

- **Risk**: The team may have limited time or technical expertise, particularly if developers are unfamiliar with Flutter or SQLite.
- **Impact**: Medium
- **Likelihood**: Medium

- **Mitigation**: Assign team members with relevant experience or allocate time for training on unfamiliar tools. Consider hiring consultants or specialists if necessary.

3. User Risks

3.1 User Adoption Issues

- **Risk**: The application may not meet users' expectations in terms of ease of use, design, or functionality, leading to low adoption rates.
- **Impact**: High
- **Likelihood**: Medium
- **Mitigation**: Perform early user testing with target users, gather feedback, and iterate on the design and functionality to ensure a smooth user experience.

3.2 Incorrect Data Input by Users

- **Risk**: Users might enter incorrect or invalid data (e.g., wrong date/time formats), causing confusion or invalid records.
- **Impact**: Medium
- **Likelihood**: High
- **Mitigation**: Implement thorough input validation on date and time fields, clear error messages, and tooltips or guides to help users enter correct data.

3.3 Privacy and Security Concerns

- **Risk**: Users may be concerned about the security of their data, especially when stored on SQLite.
- **Impact**: High
- **Likelihood**: Medium
- **Mitigation**: Implement robust security measures, such as data encryption, secure authentication, and adherence to GDPR or other relevant data protection laws.

4. Maintenance and Support Risks

4.1 Lack of Ongoing Maintenance

- **Risk**: After initial development, the app might lack proper maintenance or updates, causing issues with compatibility (e.g., new OS versions) or user dissatisfaction over time.
- **Impact**: High
- **Likelihood**: Medium
- **Mitigation**: Ensure a maintenance plan is in place for future updates and bug fixes. Assign a dedicated team or individual for ongoing support.

4.2 Compatibility with Future SQLite/Flutter Versions

- **Risk**: Future updates to Flutter or SQLite may introduce breaking changes, leading to app malfunction.
- **Impact**: Medium
- **Likelihood**: Medium

- **Mitigation**: Monitor updates to Flutter and SQLite regularly, and perform timely upgrades and testing to ensure compatibility with new versions.

5. Legal and Compliance Risks

5.1 Data Privacy Regulations

- **Risk**: Non-compliance with data privacy regulations (e.g., GDPR, CCPA) could result in legal action or fines.
- **Impact**: High
- **Likelihood**: Low
- **Mitigation**: Ensure that the app complies with relevant data protection laws, including user consent for data collection and secure data handling.

6. User Data Loss or Corruption

6.1 Database Corruption or Loss

- **Risk**: Data stored in SQLite might become corrupted or lost, leading to a poor user experience or permanent data loss.
- **Impact**: High
- **Likelihood**: Low
- **Mitigation**: Implement regular database backups and recovery strategies in SQLite. Also, ensure that the app handles database failures gracefully.

Project Progress

Feature Implementation

Week 1 (10/11/2024 – 10/13/2024): Project Planning and Setup

Summary

Focused on project planning, finalizing requirements, and setting up the development environment with the necessary tools for Flutter and SQLite.

Milestones or Risks

- **Milestone:** Finalize project requirements and set up tools.
- **Risks:** Ambiguity in requirements, tool compatibility issues.

Design

Outlined key workflows and created a document with user stories and functional requirements.

Code

No coding this week; focus was on setup.

Tests

No tests conducted.

Document

Created a Project Requirements Document and logged installation/setup details.

Week 2 (10/14/2024 – 10/17/2024): Basic UI/UX Design

Summary

Designed wireframes and low-fidelity mockups for the app's main screens, including task recording, querying, and login.

Milestones or Risks

- **Milestone:** Create initial wireframes.
- **Risks:** Design iterations could take longer than expected.

Design

Low-fidelity wireframes for:

- Task recording page
- Querying screen
- Login page

Code

No code this week; focus was on design.

Tests

Reviewed wireframes to ensure all functional requirements were represented.

Document

Added wireframes and design notes to the project folder.

Week 3 (10/18/2024 – 10/22/2024): SQLite Integration

Summary

Set up SQLite services for user authentication and task data storage.

Milestones or Risks

- **Milestone:** Integrate SQLite.
- **Risks:** Database schema design issues.

Design

Designed the SQLite schema for tasks and users.

Code

- Implemented database setup and initialization scripts.
- Wrote functions for CRUD operations in SQLite.

Tests

- Tested database setup and CRUD operations locally.

Document

Documented database schema and API functions for data handling.

Week 4 (10/23/2024 – 10/29/2024): Prototype Version 1 Development

Summary

Developed the core task recording functionality, allowing users to record tasks with date, time, and tags.

Milestones or Risks

- **Milestone:** Implement basic task recording.
- **Risks:** Ensuring smooth UI and backend integration.

Design

Integrated wireframe designs with SQLite-backed task recording forms.

Code

- Developed task recording UI and connected it to SQLite.
- Created models for task data.

Tests

- Verified tasks could be recorded and stored in SQLite.

Document

Logged task recording workflows and challenges faced during implementation.

Week 5 (10/30/2024 – 11/03/2024): User Authentication

Summary

Implemented user authentication, including login, registration, and logout features.

Milestones or Risks

- **Milestone:** Enable user authentication.
- **Risks:** Potential security flaws.

Design

Designed a login and registration flow with validation for secure user data entry.

Code

- Developed authentication endpoints.
- Integrated SQLite with login and registration forms.

Tests

- Tested user login with valid and invalid credentials.
- Verified secure data storage in SQLite.

Document

Added authentication logic documentation, including validation rules.

Week 6 (11/04/2024 – 11/08/2024): Task Query Functionality

Summary

Implemented basic task querying by date, task, and tags.

Milestones or Risks

- **Milestone:** Enable task querying.
- **Risks:** Ensuring query accuracy.

Design

Extended task recording designs to include filtering and querying options.

Code

- Added querying logic to retrieve tasks based on user criteria.
- Built a results page for displaying filtered tasks.

Tests

- Tested queries for date, task, and tag filters.

Document

Added query design and logic documentation.

Week 7 (11/09/2024 – 11/12/2024): Data Validation and Error Handling

Summary

Added input validation for task recording and querying. Implemented basic error handling for invalid or missing inputs.

Milestones or Risks

- **Milestone:** Improve data validation.
- **Risks:** Handling edge cases for invalid inputs.

Design

Defined validation rules for task input forms.

Code

- Added validation checks for date/time formats and required fields.
- Implemented error handling for invalid inputs.

Tests

- Tested task inputs with incorrect formats.
- Verified error messages displayed correctly.

Document

Documented validation logic and examples of handled errors.

Week 8 (11/13/2024 – 11/16/2024): UI/UX Finalization

Summary

Refined and polished the UI/UX based on feedback from internal reviews of the prototype.

Milestones or Risks

- **Milestone:** Finalize UI/UX for end users.
- **Risks:** Incorporating feedback may delay progress.

Design

Revised wireframes and created high-fidelity mockups.

Code

- Updated UI components to match finalized designs.
- Improved responsiveness and usability.

Tests

- Conducted usability tests with a small user group.

Document

Added updated UI/UX designs and test feedback.

Week 9 (11/17/2024 – 11/20/2024): SQLite Sync and Offline Mode

Summary

Implemented offline mode for recording tasks without a connection and syncing data when online.

Milestones or Risks

- **Milestone:** Enable offline mode and data syncing.
- **Risks:** Ensuring reliable sync without data loss.

Design

Outlined workflows for offline data entry and syncing.

Code

- Developed offline storage for tasks.
- Wrote sync logic for merging offline data.

Tests

- Tested offline task recording and syncing.

Document

Documented offline mode architecture and syncing workflow.

Week 10 (11/21/2024 – 11/26/2024): Final Version Development

Summary

Completed the final version of the application, focusing on polishing the UI and ensuring all features work seamlessly.

Milestones or Risks

- **Milestone:** Prepare the final version for review.
- **Risks:** Last-minute bug fixes.

Design

Polished UI/UX and ensured visual consistency.

Code

- Fine-tuned all features.
- Improved code efficiency and readability.

Tests

- Conducted end-to-end tests for all core features.

Document

Added final version details and polished UI/UX documentation.

Week 11 (11/27/2024 – 11/30/2024): Final Testing and Bug Fixing

Summary

Conducted rigorous testing to identify and fix remaining bugs.

Milestones or Risks

- **Milestone:** Deliver a bug-free application.
- **Risks:** Unexpected bugs delaying deployment.

Design

Minor updates based on bug fixes.

Code

Fixed reported bugs and optimized code.

Tests

- Performed extensive testing across all features.
- Verified functionality across different devices.

Document

Updated test cases and added bug fix logs.

Week 12 (12/01/2024): Deployment

Summary

Deployed the application successfully, marking project completion.

Milestones or Risks

- **Milestone:** Deploy the application.
- **Risks:** Deployment errors.

Design

No changes.

Code

Packaged and deployed the application.

Tests

Verified deployment and checked live functionality.

Document

Created deployment notes and final project report.