NOTRE DAME UNIVERSITY
LOUAIZE
جامعة سيدة اللويزة

Ekklesia

A Smart Church Management & Community App

Senior Project Presentation

Presented by

Georges El Sous

Fady Nassar

Agenda

- Background & Problem Statement
- Proposed Solution
- Use Case Diagram
- Key Features
- UI/UX Design
- Architecture
- What the AI Does
- Project Experience & Outcomes
- Conclusion

Background & Problem Statement

Many churches still use printed papers and verbal announcements to share information. Because of this old communication style, members often miss important updates, event changes, and prayer schedules.

No centralized digital platform

No personalized user experience

Difficult for members to follow church updates

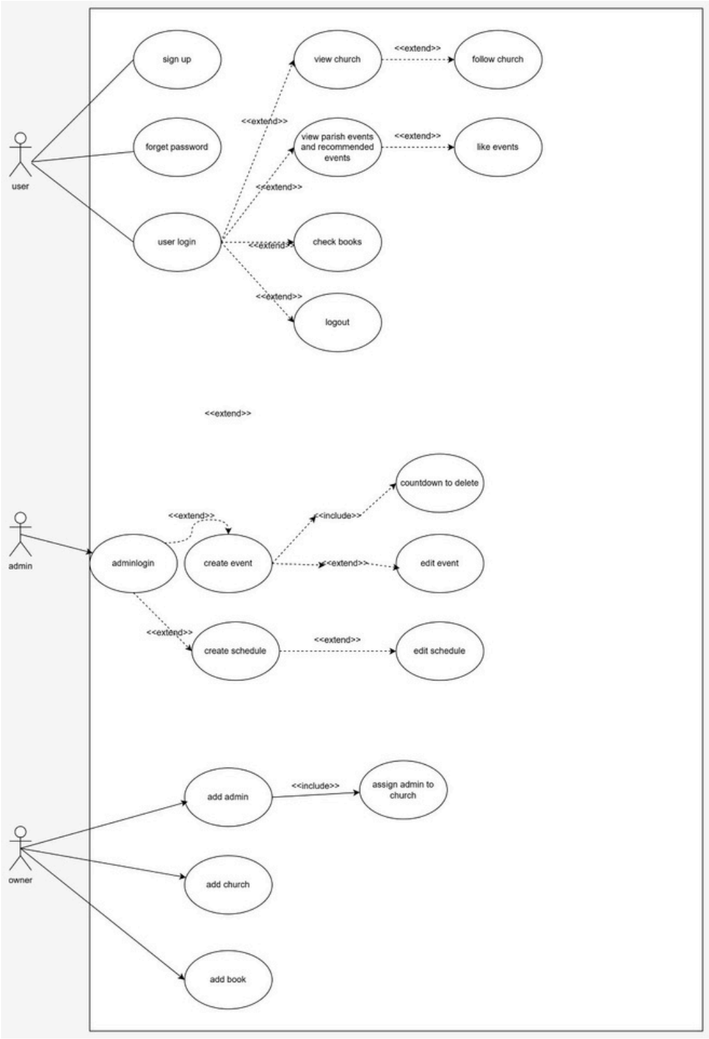
Proposed Solution

Ekklesia

A mobile app that provides:

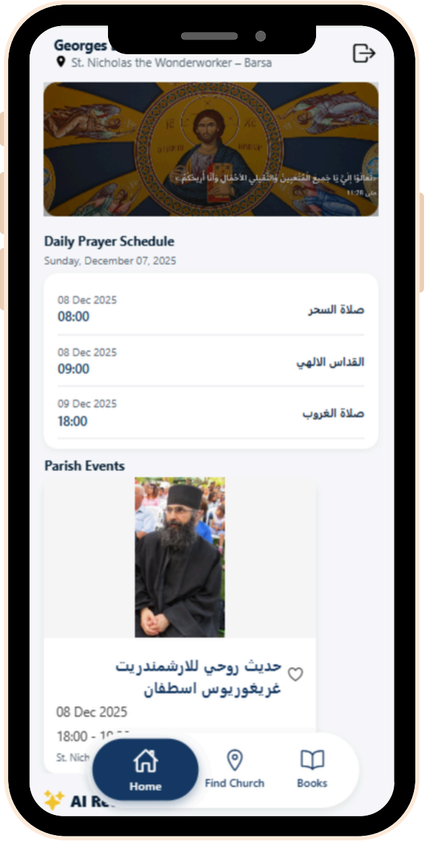
- Seamless **login & signup**
- **Event viewing, liking**, and **recommendations**
- Church following
- Admin tools to **create/manage events & schedules**
- Owner tools for managing churches, admins, and books

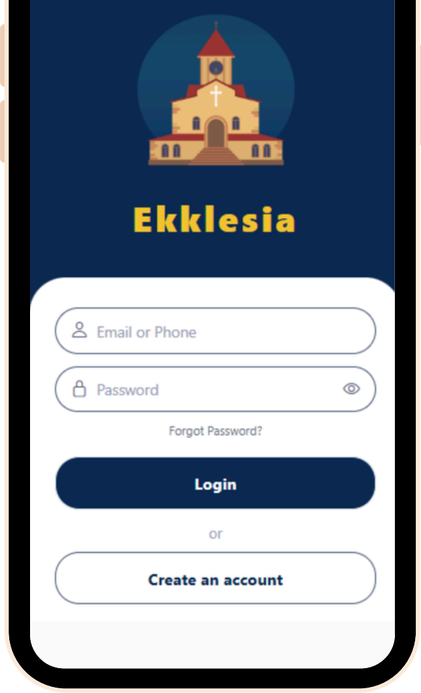
Use Case Diagram

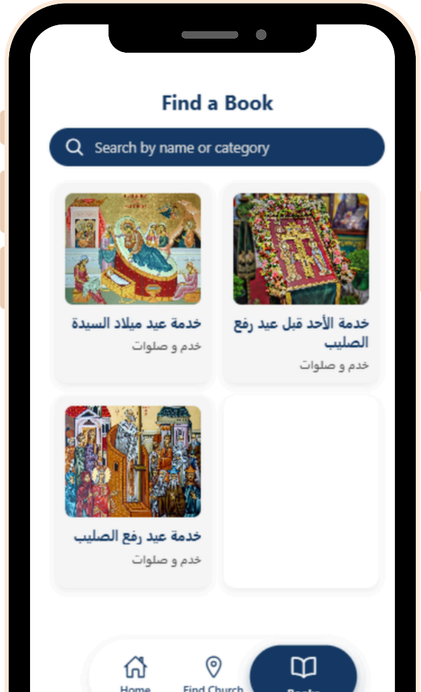


Ekklesia – Main User Features

- Sign up / Login / Logout
- View parish schedule
- View parish events
- Receive recommended events
- Like events
- Find churches
- Check available books

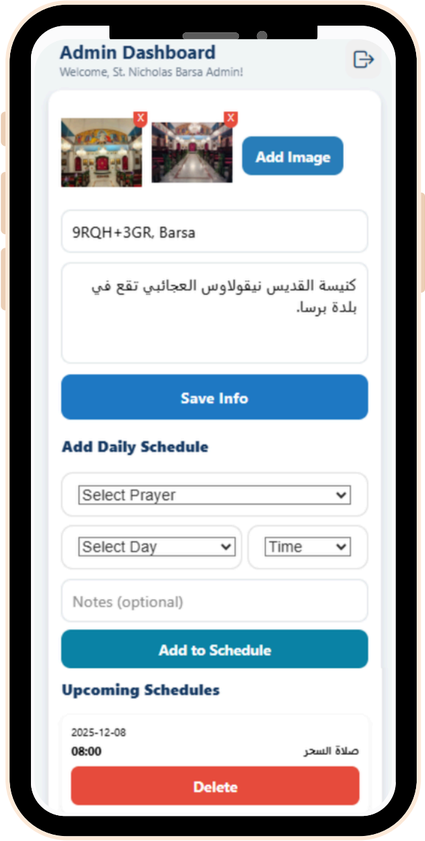


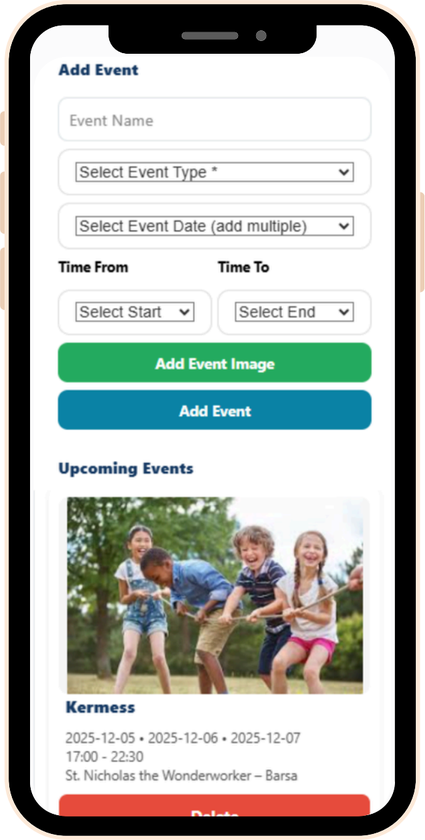




Ekklesia – Church Admin Features

- Secure admin login
- Manage church content
- Create & delete schedules
- Create & delete events

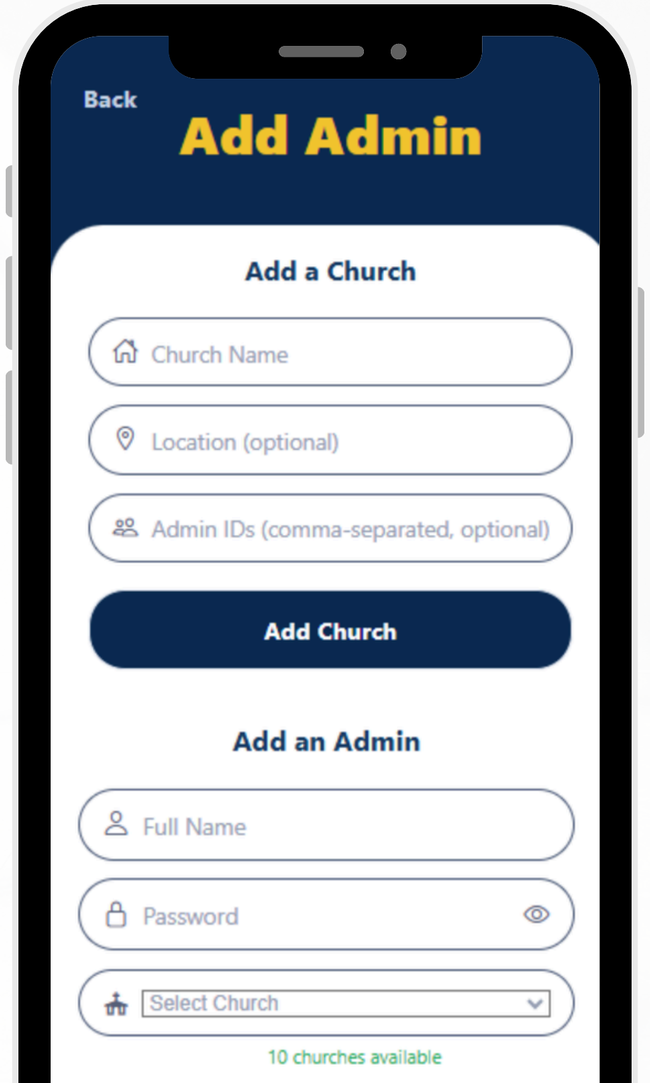




Ekklesia – Owner Features

Add church

Assign admin to church



Add admin

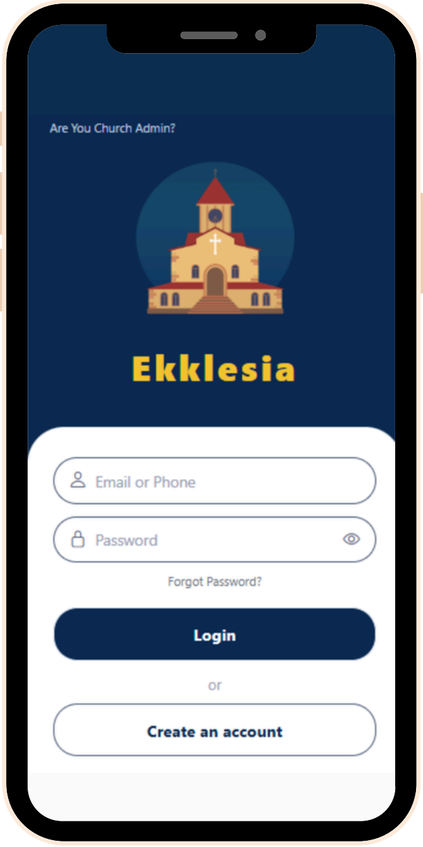
Add books to database

UI/UX Design Process

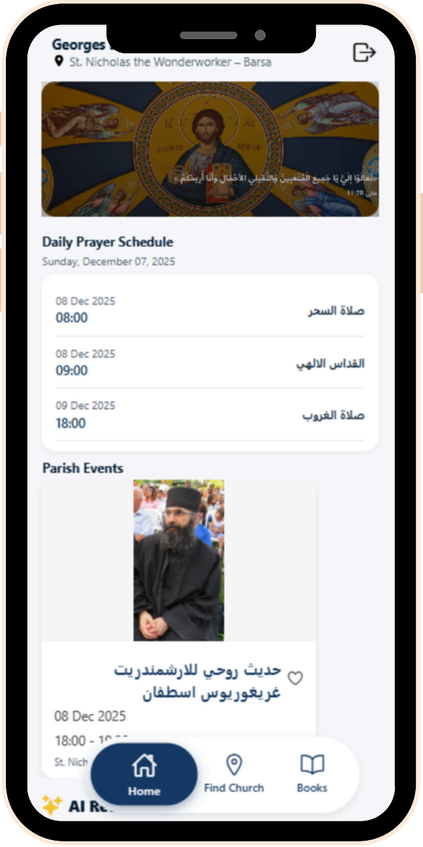
We focused on creating a clean and user-friendly interface for the Ekklesia app:

- Designed directly using **Canva**
- Clean and simple modern color palette
- Focus on **clarity and easy navigation**
- Mobile-first layout for better user experience

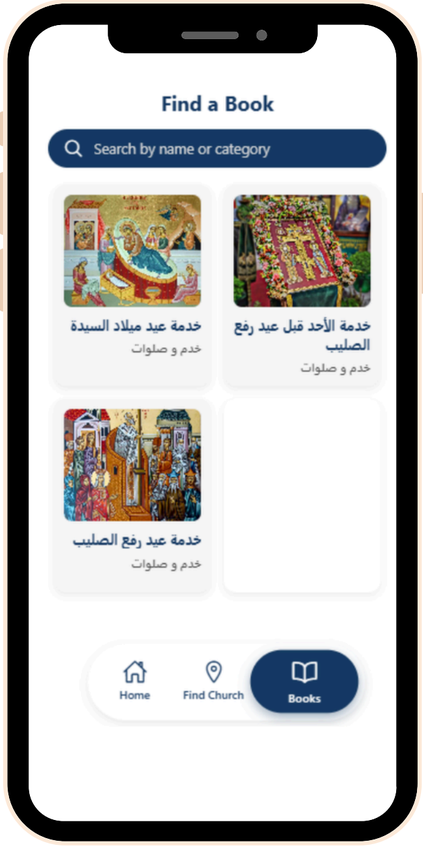
Final UI Designs



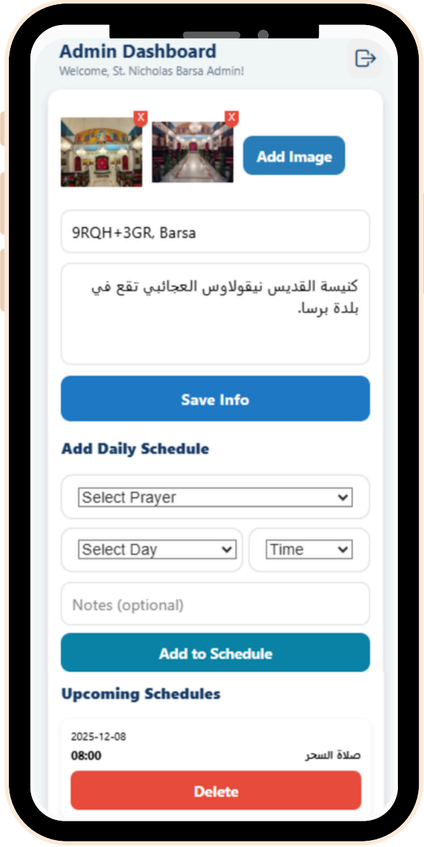
Login Page



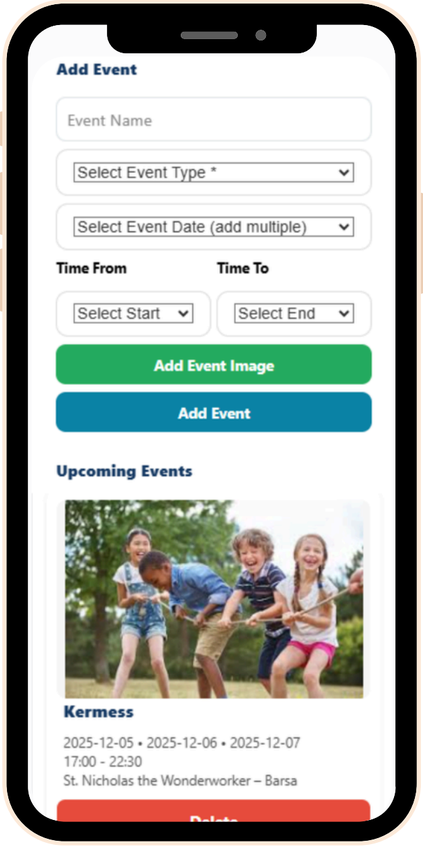
Home Page



Find Books Page



Admin Page - 1



Admin Page - 2

Architecture

Our application is built using the **MERN stack**, a modern and efficient technology setup that allows fast development, smooth performance, and easy scalability for both the mobile app and backend.

MongoDB – NoSQL database
Stores user accounts, churches, events, and books in flexible collections.

Express.js – Backend API
Handles all server routes and connects the mobile app to the database.

React Native – Mobile app
Used to build the app interface and user experience for both iOS and Android.

Node.js – Server runtime
Runs the backend code and manages all requests sent from the mobile app.

What the AI Does

- The Recommendation System uses a **scoring algorithm** to assign a score to every event.
- Higher scores mean better recommendations.
- The parameters used are how many likes an event has and how close the event is to the user's parish.
- The system also benefits from patterns learned from user behavior, similar to how modern recommendation engines compare users and items.
- When users like events, they indirectly help others by creating behavioral connections (collaborative patterns).
- After all scores are calculated, the Recommendation System sorts them and returns the **top 3 events**.

Scoring Based on Popularity

- Popular events are more likely to be **interesting for users**.
- The Recommendation System rewards popularity by adding **10 points per like** (for example, if an event has 12 likes, it gets 120 points).
- This approach follows the principle used in collaborative recommendation techniques:
- “If many users interacted with an item, it becomes valuable for others with similar behavior.”
- This value can be changed easily, making the algorithm flexible.
- The Recommendation System finds the number of likes by checking how many users added that event to their liked list.
- Popularity helps the system understand **community preferences** even when we don’t have detailed information about the event itself.

Scoring Based on Parish Match

- The Recommendation System checks if the event takes place in the **same parish as the user.**
- Events in the user's parish are usually more useful and easier to attend.
- A parish match gives a **+50-point bonus.**
- This helps local events appear higher in the recommendations, even if they are not the most liked.
- This part works like item-similarity logic: events that share important attributes with the user (like parish location) are considered a better match.
- Combining parish similarity with popularity creates a balanced and personalized recommendation.

Producing Final Recommendations

- After all events receive their scores, the Recommendation System sorts them from highest to lowest.
- The **highest-scoring events** are the “best fit” for the user.
- This ranking method is similar to how collaborative filtering systems produce final suggestions by ordering items based on learned relevance patterns.
- The algorithm takes **only the top 3 events** to recommend.
- This ensures users always see the most relevant, community supported, and personally matched events first.

What We Learned

API development and backend integration

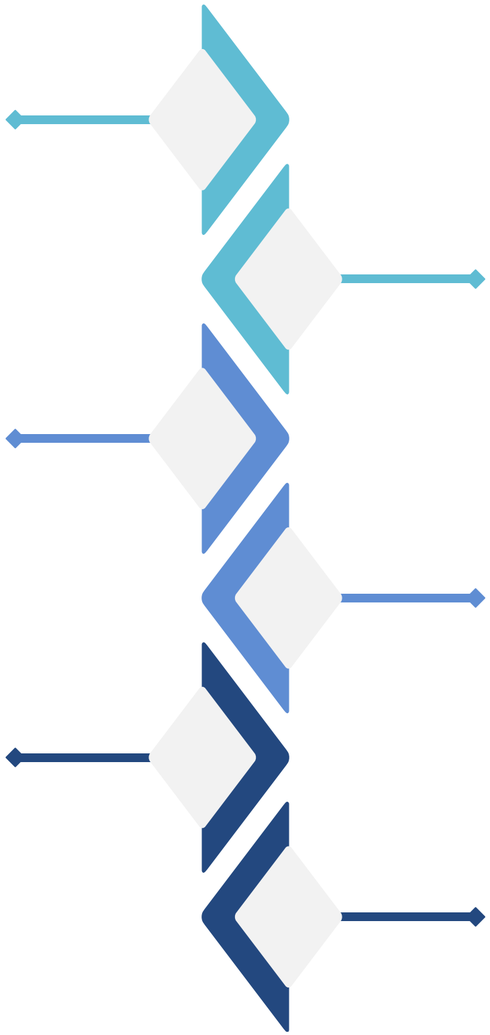
UI/UX design principles

Version control and **team** collaboration

MERN full-stack workflow

Implementing **AI** logic for recommended events

Real-world **software engineering practices**



Future Enhancements

In the next version of Ekklesia, we aim to improve the user experience by adding more powerful features that make communication easier, faster, and more personalized for every church member.

Push notifications for events

Alerts users instantly when a new event, update, or reminder is available.

Multi-language support

Makes the app usable for churches and members from different language backgrounds.

Smarter event recommendations

Improves the AI system to give more accurate and personalized suggestions.

Conclusion

- Ekklesia solves real communication challenges
- Delivers a smooth and simple user experience
- Strong technical base using MERN
- Scalable and ready for future updates

Thank You

For Your Attention