Team Competency

Rahum Burni

- **Skills:** Java, C, HTML, CSS, JavaScript, Python
- **Experience:** Completed internships in UI/UX and Data Science & AI
- **Courses Completed:**
    - Data Structures and Algorithms (CS228)
    - Intro to Object-Oriented Programming (CS227)
    - Digital Logic (CPRE281)
    - Intro to Data Science (DS201)
    - Intro to Programming in C (SE185)

Andrew Luebbers

- **Skills:** Java (primary), Python
- **Experience:** Scored 5 on the AP Computer Science A Exam, earning credits for Intro to Object-Oriented Programming (COM S 227)
- **Courses Completed:**
    - Data Structures and Algorithms (COM S 228)
    - Intro to Computer Programming in Python (COM S 127)
    - Discrete Computational Structures (COM S 230)

Ibrahim Aldulaimi

- **Skills:** Java, C, C++, Python
- **Courses Completed:**
    - Data Structures and Algorithms (CS228)
    - Intro to Object-Oriented Programming (CS227)
    - Digital Logic (CPRE281)
    - Intro to Programming in C (SE185)
- **Current Courses:**
    - Intro to Data Science (DS2010)
    - Construction of User Interfaces (SE3190)
    - Advanced Programming Techniques (COMS 3270)

Project Proposal: WiseShield

Problem Addressed: Safety

Project Description:
WiseShield is a safety-focused app with two core features:

1. **Emergency Alert:**
   Sends instant alerts to selected contacts with location details and a distress signal. The alert is triggered when the user fails to confirm their safety after a specified time beyond a set curfew. The user or their parents can set these intervals (e.g., every 30 minutes to 1 hour). This is particularly useful when the user is in an unfamiliar situation, such as being out with a stranger.
2. **Safe Tracker:**
   Allows users to share their location and walking route with trusted contacts. This feature ensures that friends, family, or roommates can monitor the user's safety when they are out, particularly in potentially risky situations like walking alone at night.

Data Management:
The app will utilize several data tables, including:

- **User Registration:**
  Users must register with a username and password. Registration will include basic information such as name, phone number, email, address, and an option for parents to register their children. The app also allows users to input their medical background, which can be sent as a PDF in case of an emergency.
- **Emergency Contacts:**
  Users can add multiple emergency contacts with options for:
  - Customized messages
  - Sharing current location (optional)
  - Sharing locations visited in the last hour at 5-minute intervals (optional)
  - Live location sharing (optional)

Alert Categories

WiseShield offers a variety of alert categories, each designed to address specific emergency scenarios and user needs. Here’s a detailed breakdown:

1. **Customized Alerts:**
   - **Pre-Configured Messages:** Users can set up personalized messages that can be sent instantly during an emergency. For instance, a message like "I'm feeling unsafe, please call me" can be triggered with a single tap.
   - **Optional Location Sharing:** These alerts can include the user's current location or recent locations visited, providing recipients with contextual information to respond effectively.
   - **Quick Access:** These alerts are easily accessible from the app’s main interface, allowing users to send them quickly in critical situations.
2. **Emergency Alert:**
   - **Broadcast to All Contacts:** This alert sends a distress message to all the user’s designated emergency contacts. The message includes the user's current location, and a log of locations visited in the past hour, updated every 5 minutes.
   - **Urgent Message:** The content of the message emphasizes the urgency of the situation, indicating that the user may be in immediate danger and requires help.
   - **One-Touch Activation:** This feature is designed to be triggered quickly, minimizing the time it takes for the user to send an alert.
3. **Medical Emergency:**
   - **Health-Specific Alerts:** Tailored for medical emergencies, this alert notifies selected contacts and local authorities that the user needs immediate medical assistance.
   - **Medical Background PDF:** The alert includes a PDF file containing the user’s medical information, such as allergies, ongoing medications, and any critical health conditions. This helps medical professionals make informed decisions upon arrival.
   - **Examples of Use:** This feature is ideal for situations like an elderly person falling or a user experiencing a severe allergic reaction. The app allows users to pre-configure these alerts based on their medical needs.
4. **Private Third-Party Alerts:**
   - **Integration with Security Services:** Users can link the app to private security companies or other third-party services they are registered with. In case of an emergency, the app can send an alert directly to these services.
   - **Customizable Alerts:** Users can create specific messages to be sent to these third-party services, which can include real-time location data or specific instructions, such as "House alarm triggered at 123 Main St."
   - **Examples of Use:** If a user's home security system detects an intruder, the app can automatically notify the security company, ensuring a rapid response.

5. **Local Authorities Alerts:**
    - **Pre-Configured by Authorities:** These alerts are set up by local law enforcement, fire departments, or other emergency services. Users cannot modify these alerts, ensuring they meet the protocols of the responding authorities.
    - **Direct Communication:** The alert sends detailed information, such as the user's address, live location, and the nature of the emergency. For example, "House fire at [Address]" or "I'm being followed, location shared."
    - **Preventing Misuse:** To avoid prank alerts, these messages are standardized and can only be sent under specific conditions defined by the authorities.
6. **Welfare Check:**
    - **Routine Check-Ins:** Users can schedule welfare checks, where the app periodically prompts them to confirm their safety. This is particularly useful for those living alone or in potentially hazardous situations.
    - **Authority-Driven Checks:** Local authorities can also initiate welfare checks if there are concerns about a user's safety. If the user does not respond within a specified time, an alert is escalated.
    - **Multiple Attempts:** The system makes several attempts to contact the user before assuming they are in distress, reducing the risk of false alarms while ensuring timely intervention if needed.

These alert categories are designed to provide comprehensive coverage for various emergency scenarios, giving users peace of mind knowing they have a reliable and responsive safety system in place.

Optional Feature/task:
Our team consist of three students as of now but if we have time or got another person to join us, we might work on a tracker. A small device where the user pushes a button which will share live location using the tracker as the baseline instead of the phone (tracker can be a small accessory for example), this helps family members/friends track the user live, this also helps local authorities to track user if needed, an example the user is a kid and got into a stranger's car and realized that they made a mistake or they got lost, they can press on the tracker which will share location, this will help local authority/friends/family members track them.

Language/Platform/libraries:

- Android/Springboot.
- Java
- API for location:

https://ipstack.com/blog/5-free-geolocation-apis-to-retrieve-user-coordinates-in-latitude-&-longitude

Large/Complex

For the **WiseShield** project, the following components are needed:

Client Side

- **Number of Screens and Complexity of Each Screen:**
    - **Login/Registration Screen:** Medium complexity, requiring user input for registration details and medical background.
    - **Home/Dashboard Screen:** Medium complexity, displaying an overview of the app's functions (e.g., emergency alert, safe tracker).
    - **Emergency Contacts Management Screen:** Medium complexity, allowing users to add/edit emergency contacts and configure alert settings.
    - **Alert Configuration Screen:** High complexity, providing options to customize and set up different types of alerts.
    - **Location Sharing Screen:** Medium complexity, showing the user's location and route, with options to share with contacts.
    - **Settings Screen:** Medium complexity, for managing user preferences, account details, and app settings.
    - **Alert History/Logs Screen:** Low complexity, displaying a history of sent alerts and their details.
- **Local Processing:**
    - Processing user inputs, validating data (e.g., registration details), and managing local tasks such as setting up alerts and notifications.
- **Networking with Server:**
    - The app will need to communicate with a backend server to send alerts, retrieve data, and sync user information across devices.
- **Connecting to APIs:**
    - Integration with external services (e.g., mapping APIs for location tracking, messaging APIs for sending alerts) is required.
- **Local Database Use:**
    - A local database is needed to store user preferences, emergency contacts, alert configurations, and temporarily cache location data before it is sent to the server.
- **Local Devices Use:**
    - The app will utilize the device's GPS for location tracking and possibly other sensors (e.g., accelerometer) to detect movement or falls.

Interaction Type

- **Turn-based, Realtime, Two-way, Peer-Peer:**
    - **Realtime Interaction:** The app requires real-time communication for location sharing and sending alerts, ensuring immediate response in emergencies.

Server Side

- **Routing:**
    - The server will handle routing for various app functions, such as user registration, login, sending alerts, and retrieving user data.
- **Server-Side Processing:**
    - The server will process incoming requests, such as verifying user credentials, managing alert dispatches, processing location data, and handling responses from emergency contacts or authorities.
- **Database Access:**
    - The server will access and manage the database, ensuring secure storage and retrieval of user information, alert histories, and location data.

Database

- **Tables # and Relationships:**
    - **User Table:** Stores user information, including registration details, preferences, and medical background.
    - **Emergency Contacts Table:** Linked to the User table, stores details of each user's emergency contacts.
    - **Alerts Table:** Stores sent alerts, including timestamp, message content, recipients, and associated user.
    - **Location History Table:** Stores location data linked to alerts, tracking the user's movements prior to and during an alert.
    - **Authorities and Third-Party Services Table:** Stores details of local authorities and private security services linked to specific users.
    - **Relationships:** The tables will have relationships such as one-to-many (e.g., one user to many emergency contacts) and many-to-many (e.g., multiple users linked to multiple authorities or third-party services).