Claudia Bridgens, Verena Tiede, Holly Farler, Aishah Hussain

1.What are you building?

We are building ‘BuddyWalk’ - an application that offers a mapping tool and buddy system connecting registered users to one another in the interest of safety when walking trips alone in the city.

2.What does it do or what kind of problem does it solve?

Responding to a growing public mood of a lack of safety when walking in the city, ‘BuddyWalk’ is designed to allow a greater security - both real and sensed - to users during trips made after dark. After registering their details, user’s will gain access to a mapping tool with which they can input their planned location, route and start time. The application then connects them to another registered user taking the same route, along with their contact details and directions to a planned meeting point. Current mapping tools often offer route advice including main roads as a safety option. As opposed to highlighting the dangers of a quieter street we aim to offer a more empowered solution. Users are not required to moderate their routines based on possible dangers but instead find safety in numbers.

3. **What are the key features of your system?**

Features are prioritised according to the MoSCoW method to help decide how to divide them between our two sprints.

‘Must-have’ features:

- **‘**Users’ database in SQL: A ‘Users’ database, written in SQL, will store the users’ information as pulled from the frontend input fields, and provide access to the location and time information needed for the find_buddy() and routing functions.

- ‘find_buddy()’ function: One of the key features of our backend will be a function that matches two users together based on the proximity of their start and end locations, and their desired journey start times. This function will pull user data from the ‘Users’ database.

- Routing function: A function that returns a route for the pair to take once matched. This will use data points already retrieved for the ‘find_buddy()’ function. The route needs to be returned to the user via the frontend.

- Python-SQL database connection: In the backend, our find_buddy() and routing functions will require a connection to our ‘Users’ database, as will our API.

- Our API: We will write our own API to push outputs from our find_buddy() and routing functions to the user, and to pull inputs for start and end location and journey start time. API endpoints must include a landing page, input pages, and an output page.

- Connection to a mapping API: For the functions mentioned above, to match buddies by distance, and to return them a suitable route, we will require a connection to a free mapping API. A possible candidate is the Google Maps Distance Matrix API (https://developers.google.com/maps/documentation/distance-matrix/overview).

‘Should-have’ features:

- Simple, appealing frontend: We should be able to provide a pleasant user experience by coding a simple frontend design, for example using React alongside Flask.

‘Could-have’ features:

- Map embedding: In order to improve user experience, the Google Maps Embed API (https://developers.google.com/maps/documentation/embed/get-started#search), allows for map embedding on our frontend. This could be used to allow location selection by the user, as well as to display the calculated route.

- Within the broader task of User Management, a good ‘could-have’ feature would be a user registration functionality, which would in theory allow for the re-use of our web app, and of some kind of user security authentication (a ‘won’t-have’ for our system).

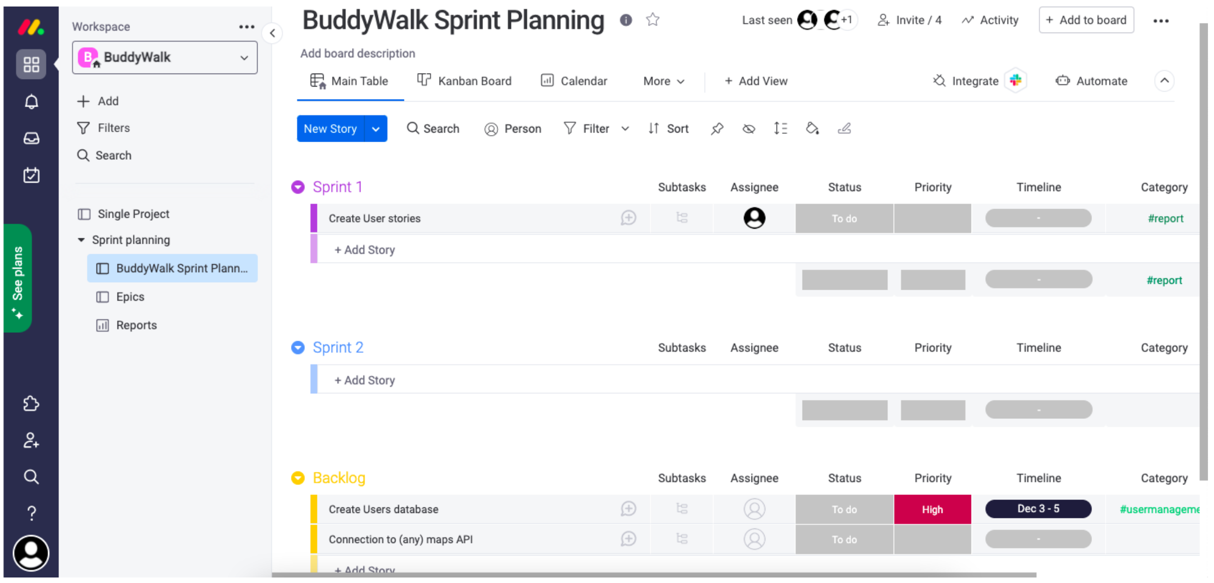
- If users are registering, we would also require a user login functionality, for which we could potentially use login.py. These features would interact with the ‘Users’ database: a registration feature would populate the database, and the login feature would refer to it to verify login details.

‘Won’t-have’ features:

- We recognise the potential need for a means of authenticating users of the app in terms of their identity and safety to others. As this is outside the scope of our project, our web app will not include such a feature, but it would be our intention in a real-world scenario.

5. **Describe the team approach to the project work**: how are you planning to distribute the workload, how are you managing your code, how are you planning to test your system.

In order to maintain an overview over what parts our project consists of, every member’s responsibilities, as well as our own deadlines, we set up a sprint planning board on monday.com. In an agile way, it consists of a backlog section and two sprint sections. Each sprint will last for a week, starting this Saturday 4th December and finishing Saturday 18th December, one day before the deadline. In our first sprint planning meeting this week Saturday, we will determine all items in the backlog and discuss what we want to accomplish in our first sprint. We will create user stories for potential users of our application.



Workspace

BuddyWalk

Add · Filters · Search

Single Project

Sprint planning

- BuddyWalk Sprint Plann...
- Epics
- Reports

BuddyWalk Sprint Planning

Add board description

Main Table · Kanban Board · Calendar · More · Add View · Integrate · Automate

New Story · Search · Person · Filter · Sort

Sprint 1

	Subtasks	Assignee	Status	Priority	Timeline	Category
Create User stories			To do		-	#report
+ Add Story						
					-	#report

Sprint 2

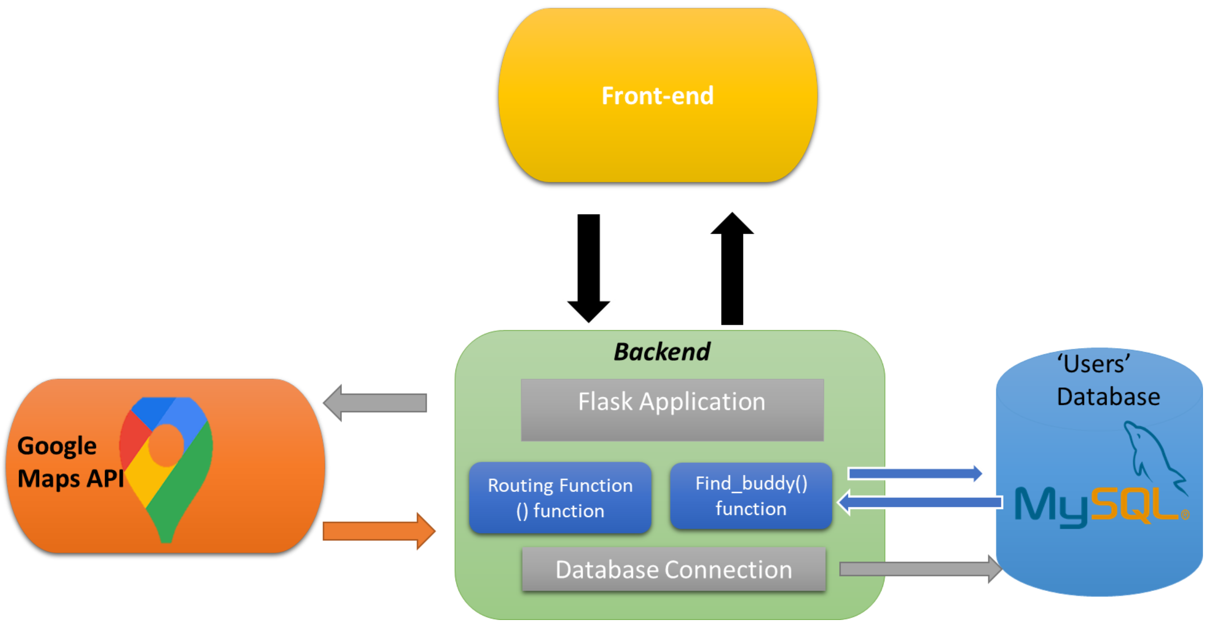
	Subtasks	Assignee	Status	Priority	Timeline	Category
+ Add Story						
					-	

Backlog

	Subtasks	Assignee	Status	Priority	Timeline	Category
Create Users database			To do	High	Dec 3 - 5	#usermanageme
Connection to (any) maps API			To do		-	
+ Add Story						

The work is divided as follows, following everyone's individual strengths: Claudia will work on the user database and connecting it to our application, Aishah will work on the maps integration and route calculations, Verena will work on the api and Holly will work on the code that matches users. Everyone will work on their own branch of the same Github repo, and add everyone as a reviewer for pull requests. Everyone will be responsible for writing the unit tests for their own part of the program. Finally, we will perform UAT from the point of view of the users we created user stories for.

4. Provide a sample architecture diagram of your system (you can use PPT with squares and circles to demonstrate a simplified flow of your system)


Front-end
Backend
Flask Application
Routing Function () function
Find_buddy() function
Database Connection
Google Maps API
'Users' Database
MySQL