**Functional requirements**

**Functional Requirement FR-001: User Registration.**

**Description**: The system shall allow users to create an account by providing a valid email address and password to access the platform.

* User inputs: user email address and password.

**Acceptance Criteria:**

* Users should be able to provide a valid email address and password during registration.
* The system should validate the uniqueness of email addresses to prevent duplicate accounts. If the entered information is not valid, the system should show an error message "The entered email address or password is not valid".

**Functional Requirement FR-002: User Login.**

**Description**: Users shall be able to log in into the system using their registered email and password.

* User inputs: user email address and password.

**Acceptance Criteria:**

* Users should enter a valid email and the correct password to access the platform.
* The system shall verify user credentials and grant access upon successful authentication.
* If the entered information is not valid, the system should show an error message, "The entered email address or password is not valid."

**Functional Requirement FR-003: Log out.**

**Description**: The system shall allow users to log out of the platform.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.

**Functional Requirement FR-004: General Information.**

**Description**: The system shall provide users with general information about Sydney, such as transportation, emergencies, weather, the process at the airport, and useful links.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.

**Functional Requirement FR-005: Add expenses calculator.**

**Description**: The system shall provide users with an expense calculator template to help them calculate their expenses.

* User inputs: expenses description and cost.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.

**Functional Requirement FR-006: Edit expenses calculator.**

**Description**: The system shall allow users to edit the expense calculator template.

* User inputs: expenses description and cost.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* At least one expense must be added.

**Functional Requirement FR-007: Add check list.**

**Description**: The system shall provide users with a check list creator to help them carry the most important stuff before heading into a specific place.

* User inputs: item name, item description.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.

**Functional Requirement FR-008: Edit check list.**

**Description**: The system shall allow users to edit a check list.

* User inputs: item name, item description.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* The check list must be created already.

**Functional Requirement FR-009: Offering tourist places.**

**Description**: The system shall provide users with tourist places in Sydney based on their interests.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* Users’ interests should be chosen from the multiple interests’ options.

**Functional Requirement FR-010: Add tourist plan.**

**Description**: The system shall allow users to add a tourist plan from the places offered by the system based on their interest in organising the trip.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* Users’ interests should be chosen from the multiple interests’ options.
* Users should pick at least one place to add a plan.

**Functional Requirement FR-011: Remove tourist plan.**

**Description**: The system shall allow users to remove a tourist plan.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* Users should already be on a specific plan.

**Functional Requirement FR-012: Edit tourist plan.**

**Description**: The system shall allow users to Edit a specific tourist plan by changing some places with others or remove it from the plan entirely.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* Users should be already in a specific plan.

**Functional Requirement FR-013: User feedback.**

**Description**: The system shall allow users to give feedback about the platform regarding issues to be fixed, suggest ideas, or give positive feedback by filling out a feedback form.

* User inputs: Name, email address, feedback type and message.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.

**Functional Requirement FR-014: Contact us.**

**Description**: The system shall allow users to contact the owner of the platform by filling out the Contact Us form.

* User inputs: Name, email address, contact number and message.

**Acceptance Criteria:** No acceptance criteria

**Functional Requirement FR-015: Create profile.**

**Description:** The system shall allow users to create profiles with personal information such as name, age, photo, bio, and gender.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.

**Functional Requirement FR-016: Edit profile.**

**Description:** The system shall allow users to edit profile information such as name, age, photo, bio, and gender.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* A user profile must be created first.

**Functional Requirement FR-017: Create group.**

**Description:** The system shall allow users to create a group trip for a specific plan to allow other users to join in.

* User inputs: group name, the limit number of users, allowed age group.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.

**Functional Requirement FR-018: Remove group.**

**Description:** The system shall allow users to remove a group trip from a specific plan.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* The creator of the group is able to remove a specific group, even if there are many users in it.
* Group must be created first.

**Functional Requirement FR-019: Edit group specification.**

**Description:** The system shall allow users to edit a group trip specification for a specific plan, such as group name, limit number of users, and allowed age group.

* User inputs: group name, the limit number of users, allowed age group.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* Group must be created first.

**Functional Requirement FR-020: Show groups.**

**Description**: The system shall be able to show users groups in a page to join in.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* At least one group should be created, or the system should show a message "There are no groups".

**Functional Requirement FR-021: Search about groups.**

**Description**: The system shall be able to allow users to search for groups to join in.

* User inputs: group name, gender, age and date.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.
* At least should be one group created.

**Functional Requirement FR-022: Connect with google API.**

**Description**: The system shall connect with the Google API to retrieve touristy place information such as pictures, rates, and contact information.

**Acceptance Criteria:**

* Users should have been logged into the platform previously.

**Non-functional requirements**

**Availability:** The system should be available 99.99% of the time during normal business hours (8 AM to 8 PM, Monday through Friday) and 99.95% of the time on weekends and holidays.

**Scalability:** The system should be able to manage a 20% increase in concurrent users within five minutes after the load has increased.

**Response Time:** Under normal operating conditions, the system should respond to user interactions (such as button presses) within 300 milliseconds.

**Data Security:** Using industry-standard encryption algorithms, all confidential user data (such as passwords and personal information) must be encrypted both at rest and during transmission.

**User Interface Responsiveness:** The user interface should be responsive and functional on devices with different screen sizes, such as tablets and smartphones.

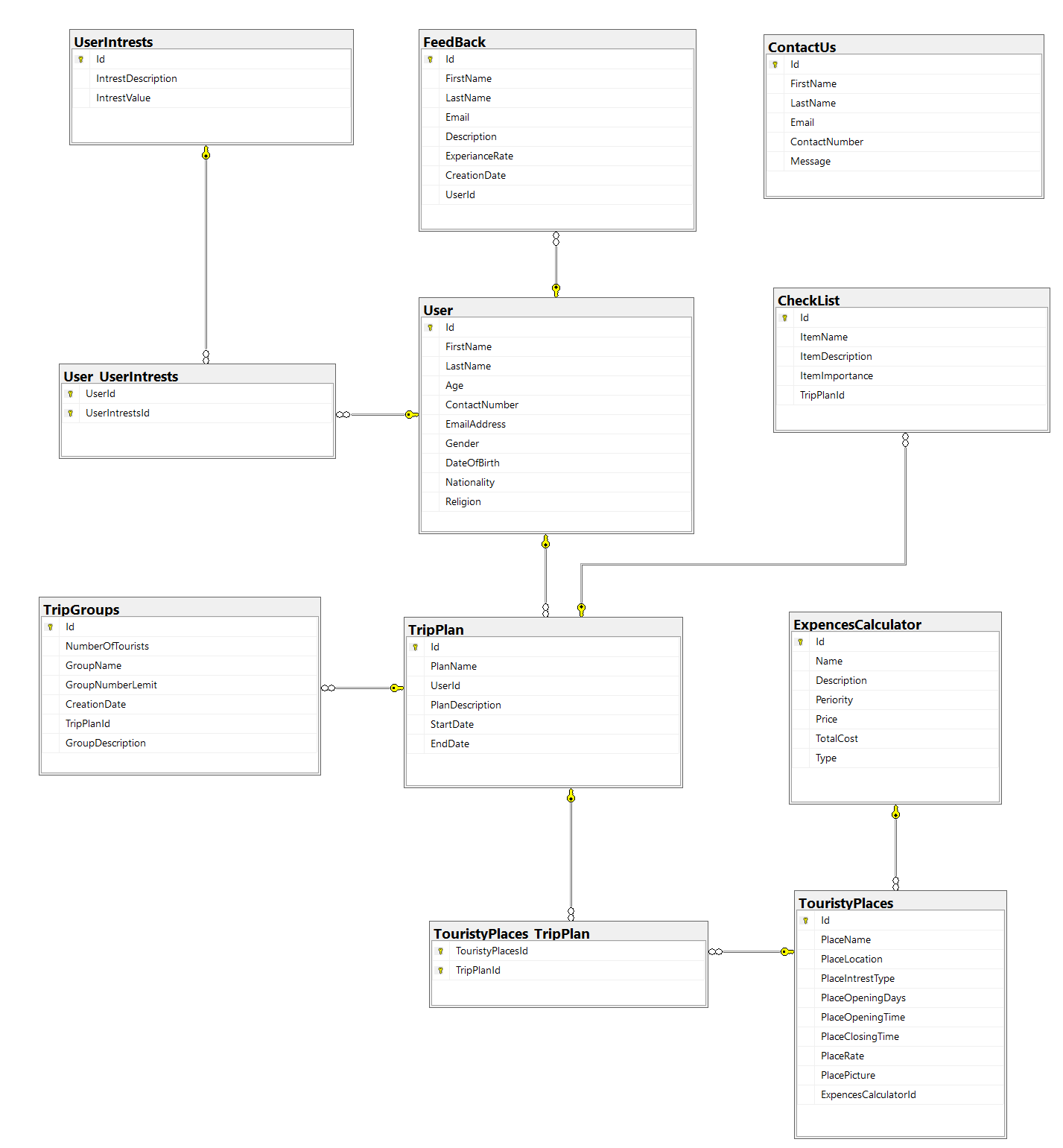
**Reliability:** The system should be able to recover from a failure in less than five minutes and recommence normal operations with no data loss.

**Load Handling:** The system should be able to support a thousand concurrent users with no breaking 80% of its available resources.

**Maintainability:** The codebase should adhere to coding standards and contain thorough documentation, making it simple for new developers to comprehend and maintain the system.

**Accessibility:** The user interface should adhere to WCAG (Web Content Accessibility Guidelines) level AA standards, to guarantee it is accessible to users with disabilities.

**ERD diagram**



**Tasks allocation**

|  |  |
| --- | --- |
| **Member name** | **Allocated tasks** |
| **Azzam Nasser Althunayyan** | **6 tasks** |
| **SK** | **5 tasks** |
| **Jianbian** | **5 tasks** |
| **Calvin** | **5 tasks** |